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. Most existing privacy preserving techniques are designed to resist semi-honest adversaries and require intense computation to perform data analysis. Secure collaborative learning is significantly difficult with the presence of malicious adversaries who may deviate from the secure protocol. Another challenge is to maintain high computation efficiency with privacy protection. In this paper, matrix encryption is applied to encrypt data such that the secure schemes are against malicious adversaries, including chosen plaintext attack, known plaintext attack, and collusion attack. The encryption scheme also achieves local differential privacy. Moreover, cross validation is studied to prevent overfitting without additional communication cost. Empirical experiments on real-world datasets demonstrate that the proposed schemes are computationally efficient compared to existing techniques against malicious adversary and semi-honest model.

Index Terms—Malicious adversary, local differential privacy, chosen plaintext attack, linear model.

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 features on the same sets of samples. 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 features for disjoint sets of samples. In many cases data are collected over different sites with the same features. 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 coopetitive 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 scheme is resilient to malicious adversaries, including chosen plaintext attack, known plaintext attack, and collusion attack which compromises all but one agency. If any agency deviates from the scheme, the result is not accurate but still no sensitive information of original data is disclosed.

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

4) The scheme has high computational efficiency to analyze large datasets with high accuracy.

The rest of the paper is organized as follows. Section II reviews the related work. Preliminaries are presented in Section III. In Section IV, we provide the system overview.
Section V introduces the proposed scheme. Security analysis is given in Section VI. Section VII provides the performance evaluations by simulation. Finally, Section VIII concludes the paper.

II. RELATED WORK

Techniques for privacy preserving data analysis fall 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 encryption 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 encryption 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 encryption 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 encryption 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 samples (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. GuardLR [25] is another secure linear regression designed to against malicious adversary. GuardLR requires two cloud servers with one for secure training and another for secure prediction. The collusion between these two cloud servers is not allowed.

Table I summarizes main references studying privacy preserving linear regression 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                  |
| [26]           | Yes      | No                  |
| [27]           | Yes      | No                  |
| [16]           | No       | No                  |
| [24]           | No       | No                  |
| [5]            | No       | No                  |
| [3]            | No       | Yes                 |
| [2]            | No       | Yes                 |
| [25]           | Yes      | Yes                 |

Table I: Related work of privacy preserving linear regression 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).
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 [28]. 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 (DP) has also been investigated [29]–[34]. [29] enforced DP by perturbing the objective function of the optimization problem. [31] reduced the dimension of features while [32] reduced the dimension of samples for DP. [33] built an local differential privacy-compliant stochastic gradient descent algorithm. [34] designed univariate linear regression (i.e., model only includes one feature) with DP. All these studies focused on analyzing single dataset instead of collaborative learning. Moreover, dimension reduction (either dimension of samples or features) disables cross validation or deriving model estimates for each feature.

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

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

where $Y \in \mathbb{R}^n$ is a vector of responses, $X \in \mathbb{R}^{n \times p}$ is the feature matrix, $\beta$ is a $p \times 1$ vector of regression coefficients, $e$ is an $n \times 1$ vector of random errors and $\mathcal{N}$ denotes multivariate normal distribution. $n$ is the number of samples and $p$ is the number of features. The estimate for $\beta$ is

$$\hat{\beta} = (X^TX)^{-1}X^TY.$$

Ridge regression is widely used to do variable selection for high-dimensional datasets [35]. It minimizes the residual sum of squares subject to a bound on the $L_2$-norm of the coefficients

$$\hat{\beta}_{ridge} = \underset{\beta}{\text{argmin}} \{(Y - X\beta)^T(Y - X\beta) + \lambda \beta^T \beta\}.$$

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

B. 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, we have

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

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 model using all the data while preserving data confidentiality. Suppose there are $K$ agencies and agency $i$ has $X_i$ and $Y_i$ ($i = 1, \ldots, K$). We focus on horizontal partitioning scenario where agencies have the same set of features and different sets of samples.

The proposed privacy preserving scheme contains pre-modeling, modeling and post-modeling phase (Figure 1). Data are encrypted in the pre-modeling phase. Encrypted data are then sent to cloud computing service provider (i.e., cloud server). The cloud server can be any agency participating in the collaborative learning. In the modeling phase, the cloud server conducts privacy preserving linear model using encrypted data. The cloud server then sends encrypted model results back to agencies. The encrypted model results are decrypted in the post-modeling phase.

B. Threat model

Assume that adversary model is malicious, i.e., agencies or the cloud server may arbitrarily deviate from the scheme specification and use any efficient attack strategy. The malicious adversary can be classified into the following categories.

- The malicious agencies may generate fake data and conduct chosen plaintext attack [36] to recover the encryption matrices generated by other agencies.
- If part of the original data is disclosed, malicious adversary can recover private data by known plaintext attack [37].
- The agencies and cloud server may execute different computation than expected.
- The cloud server may intentionally return a random or forged result.
- A collusion attack may compromises any malicious agency or cloud server.

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 [38]). However the proposed scheme ensures that once an agency provides an input into the computation, the agency is bound to using the same input consistently throughout the entire computation.
C. Design goals

The design goals are summarized as follows.

- Privacy: Any agency may perform maliciously. A strong threat model includes collusions which compromise all but one agency, cloud server, and adversary outside the collaborative learning. We design encryption method resilient to malicious behavior of agencies and cloud server, including collusion attack, chosen plaintext attack, and known plaintext attack.
- Soundness: The privacy preserving scheme is able to verify if all the agencies and cloud server behave honestly.
- Efficiency: The secure scheme is computationally efficient and achieves high accuracy.

V. THE PROPOSED SCHEME

A. Pre-modeling phase

We first describe the encryption method for a single agency. To simplify the notation, we use X as feature matrix and Y as response. Additive noise $\Delta$ is added to original dataset in the first layer, i.e., $\tilde{X} = X + \Delta$. In the second layer, data is further encrypted by row and column transformation, i.e., $X^* = AXB$ ($A$ and $B$ are randomly generated orthogonal and invertible matrix, respectively). To summarize, the encryption mechanism $g$ is $g(X) = (X + \Delta)B$. For response $Y$ encryption, two pseudo responses are generated with one for verification and another to enhance encryption. Define the first pseudo response $Y_{s1} = \sum_{i=1}^{p} x_i$ where $x_i$ is the $i$-th column in $X + \Delta$. The second pseudo response $Y_{s2}$ is generated randomly. Let $Y^* = A[Y_{s1}, Y_{s2}]C$ where $C$ is a $3 \times 3$ random invertible matrix. The detailed encryption approach of each agency is given as follows.

1) First layer encryption

Agency $i$ generates noise matrix $\Delta_i$ with each element following Gaussian distribution $N(0, \sigma^2)$. $\Delta$ is added to $X_i$ to get encrypted data $\tilde{X}_i = X_i + \Delta_i$.

2) Second layer encryption

In order to maintain data utility for linear model, $\tilde{X}_i (i = 1, 2, \cdots, K)$ needs to be encrypted by identical $B$ in the final encryption data since the aggregated dataset is in the form of $[\tilde{X}_1^T, \tilde{X}_2^T, \cdots, \tilde{X}_K^T]^T$. Because each agency does not know encryption matrices generated by other agencies, we encrypt $\tilde{X}_i$ by all agencies with the commutative encryption matrix $B_i$ generated by agency $i$. This specific encryption approach guarantees that $\tilde{X}_i$ is encrypted by identical $B = \prod_{i=1}^{K} B_i$.

Agency $i$ generates orthogonal matrix $A_{i1}, A_{i2}, \cdots, A_{iK}$, invertible matrix $B_i$ and $3 \times 3$ invertible matrix $C_i$ ($i = 1, 2, \cdots, K$). Suppose agency $i$ has $n_i$ samples and $p$ features. To make $B_i$ commutative, each agency first generates $p \times p$ dimensional $B_0$ locally as the matrix basis using the same random seed. Then agency $i$ generates a vector of random coefficients $(b_{i1}, \cdots, b_{ip})$ and computes $B_i = \sum_{j=1}^{p} b_{ij} B_0$. Similarly, agency $i$ generates common invertible matrix basis $C_0$, a random vector $(c_{i1}, c_{i2}, c_{i3})$, and compute $C_i = \sum_{j=1}^{3} c_{ij} C_0$. $B_i$ has dimension $p \times p$, $C_i$ has dimension $3 \times 3$, and the dimension of $A_{i1}, A_{i2}, \cdots, A_{iK}$ is $n_i \times n_i, n_i \times n_i, \cdots, n_K \times n_K$, respectively ($i = 1, 2, \cdots, K$). Agency $i$ generates pseudo responses $Y_{s1i}$ and $Y_{s2i}$. Let $X_{i}^* = A_{i1} X_i B_i$ and $Y^* = A_{i2}[Y_{s1i}, Y_{s2i}]C_i$. Agency $i$ releases $X_{i}^*$ and $Y^*$ to other agencies. Agency $j$ ($j \neq i$) encrypts received data and releases $A_{j1} X_{i}^* B_j$ and $A_{j2} Y^* C_j$. $X_{i}^*$ and $Y^*$ are encrypted by all agencies in a pre-specific order.

Algorithm 1: Pre-modeling phase

Input: $p \times p$ invertible matrix $B_0$ and $3 \times 3$ invertible matrix $C_0$

Output: Encrypted feature matrix and response

for Agency $i = 1, 2, \cdots, K$ do

generate a $p$-dimensional random vector $(b_{1n}, \cdots, b_{pn})$, a $3 \times 3$ random vector $(c_{1n}, c_{2n}, c_{3n})$, noise matrix $\Delta_i$ following $N(0, \sigma^2)$ and orthogonal matrices $A_{1n}, A_{2n}, \cdots, A_{Kn}$ with dimension $n_1 \times n_1, n_2 \times n_2, \cdots, n_K \times n_K$, respectively;

$B_i = \sum_{j=1}^{p} b_{ij} B_0, C_i = \sum_{j=1}^{3} c_{ij} C_0$.

for Agency $i = 1, 2, \cdots, K$ do

generate $Q_i$, a permutation of $\{1, \cdots, K\}$ with $Q_i(1) = i$;

generate $X_i = X_i + \Delta_i$, let the first pseudo response $Y_{s1i}$ be the sum of all columns in $X_i$;

generate the second pseudo response $Y_{s2i}$ randomly;

compute $X_{i}^* = A_{i1} X_i B_i$,

$Y^* = A_{i2}[Y_{s1i}, Y_{s2i}]C_i$ and send to $Q_i(2)$;

$j = 2$;

while $j \leq K$ do

Agency $Q_i(j)$ compute

$X_{i}^* = A_{Q_i(j)}(1) X^* B_{Q_i(j)}$ and

$Y^* = A_{Q_i(j)}(2) Y^* C_{Q_i(j)}$;

send $X_{i}^*$ and $Y^*$ to agency $Q_i(j + 1)$;

$j = j + 1$;

return $X_{i}^*$ and $Y^*$;

Algorithm 1 gives detailed encryption procedures.

Suppose the communication order among agencies is $1 \rightarrow 2 \rightarrow \cdots \rightarrow K \rightarrow 1$, the final released dataset would be $X^* = AXB$ where $X = [X_1^T, X_2^T, \cdots, X_K^T]^T$,

\[
A = \begin{bmatrix}
A_{K1} & \cdots & A_{Ki} & \cdots & A_{K2} & \cdots & A_{K1} \\
\vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
A_{K1} & \cdots & A_{K2} & \cdots & A_{K1} & \cdots & A_{K2} \\
\end{bmatrix}
\]

and $B = B_1 B_2 \cdots B_K$ as $B_i$ ($i = 1, 2, \cdots, K$) is commutative. Let $Y_n$ be the aggregated response data with 3 columns where the first column is the responses of $K$ agencies (i.e., $[Y_{s11}^T, Y_{s21}^T, \cdots, Y_{s1K}^T]^T$), the second column is the first pseudo response of $K$ agencies (i.e., $[Y_{s12}^T, Y_{s22}^T, \cdots, Y_{s1K}^T]^T$), and the third column is the second pseudo response of $K$ agencies.
agencies (i.e., $[Y_{121}^T,Y_{222}^T,\cdots,Y_{s_K}^T]^T$). Then $Y^* = AY_aC$
where $C = C_1C_2\cdots C_K$.

The computation complexity of encryption matrix generation and multiplication increases when the dimension of dataset increases. For dataset with big number of samples, we partition orthogonal encryption matrix $A_{ii}$ into block diagonal matrix to improve computation efficiency. It is the same to partition $A_i$ since $A_{ii}$ is generated using matrix basis $A_0$. For example, there are 10,000 samples in $X_i$. If $A_0$ is partitioned with block size 100, agency $i$ generates 100 orthogonal matrices with dimension $100\times100$ instead of one $10,000\times10,000$ matrix. In other words, $A_{ii} = diag(A_1,\cdots,A_{100})$ where $A_i$ ($i = 1,\cdots,100$) are random orthogonal matrices. The same strategy can be applied when the dimension of features is big and agencies use partitioned block diagonal matrix $B$ as encryption matrix.

B. Modeling and post-modeling phase

We first analyze model results for special case $\Delta = 0$. The encrypted response is $Y^* = AY_aC$ where $Y_a$ contains the true response and two pseudo responses. $\beta^* = (X^*T X^*)^{-1} X^*T Y^* = B^{-1}\hat{\beta}C$. $\beta$ is a $p \times 3$ matrix with the first column being the true estimate, the second column corresponding to the estimate for the first pseudo response and the third column being the estimate for the second pseudo response. To compute $\beta$, $\beta^*$ is sent to each agency in order to eliminate $B = \prod_{i=1}^K B_i$ and $C = \prod_{i=1}^K C_i$. The order of the agency decryption can be random since $B_i B_j = B_j B_i$ and $C_i C_j = C_j C_i$. Detailed procedure of $\beta^*$ decryption is given in Figure 2.

For $\Delta \neq 0$, $\beta_\Delta^* = B^{-1}((X + \Delta)^T (X + \Delta))^{-1}(X + \Delta)^T Y_a C$. The decryption approach is the same as given above. After decryption, $\beta_\Delta = ((X + \Delta)^T (X + \Delta))^{-1}(X + \Delta)^T Y_a$ is a $p \times 3$ matrix with the first column being the true model estimate.

Privacy preserving ridge regression: In order to compute $\hat{\beta}_{ridge}$, matrix $B^T B$ needs to be computed and released additionally from pre-modeling phase using similar procedures in Algorithm 1. For a given $\lambda$, we have $\hat{\beta}_{ridge} = \left( X^*T X^* + \lambda (B^T B)\right)^{-1}X^*T Y^* = B^{-1}\hat{\beta}_{ridge} C$ from encrypted datasets. So $\hat{\beta}_{ridge} = B\hat{\beta}_{ridge} C^{-1}$. Follow the decryption procedure in Figure 2 to get $\hat{\beta}_{ridge}$. The pseudo response for ridge regression is different from those used in linear regression. Since different $\lambda$ yields different model estimates using previously defined pseudo response $Y_{s_1} = \sum_{i=1}^p x_i$, let the new $Y_{s_1}$ be a vector of 0’s for ridge regression to against malicious adversary.

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 above. For $k$-cross validation, each agency uses $k$-blocked orthogonal matrix to encrypt data and samples within each block are used as training or testing set. Our schemes are efficient and practical to change regularization parameter $\lambda$ while previous secure ridge regression models used public and fixed $\lambda$ [2], [3], [6], [23].

VI. Security analysis

Without loss of generality, we use $X$ to denote the original data, and $\Delta$, $A$, $B$ to denote additive noise matrix, orthogonal and invertible encryption matrices, respectively.

Matrix encryption has been widely used in previous privacy preserving studies. $X + \Delta$ has been widely used as encryption method but has disclosure risks demonstrated in previous literatures [39]-[42]. Privacy methods with encrypted matrix in the form of $AXB$ and $AX$ were studied in [11]-[13], [15]. $XB$ and $AX$ have high data utility but often face known plaintext attack (aka, known input-output attack) and chosen plaintext attack [36], [37]. Sparse matrix encryption in the form of $AXB$ ($A$ and $B$ are sparse matrices) were investigated with enhanced privacy in [16]-[20]. The privacy guarantee needs to be argued carefully for the sparse matrix encryption as discussed in [18], [19]. Matrix encrypted in the form of $AXB$ has been proved to solve different problems without releasing sensitive information in previous works [17]-[21].

In this paper, we encrypt data from both left side and right side using dense orthogonal/invertible matrices. The framework of privacy analysis against malicious adversary is as follows.

- detection of agencies’/cloud server’s malicious computation;
- security of encryption matrix $B \rightarrow$ resilience of chosen plaintext attack;
- Security of $X$;
  - partial prior information of $X$ disclosed $\rightarrow$ resilience of known plaintext attack;

![Fig. 2: Post-modeling procedure.](image-url)
- local differential privacy;
- Resilience to malicious adversary.

\section*{A. Resilience to malicious adversary}

\subsection*{1) Detection of agencies’cloud server’s malicious computation}

The first pseudo response $Y_{s1}$ is included in the encrypted response matrix $Y_a$ to verify the output received from the cloud (Figure 3). Since $Y_{s1} = \sum_{i=1}^{p} x_i$, where $x_i$ is the $i$-th column in $X + \Delta$, $\beta_{s1}$ (the second column in model estimate matrix $\hat{\beta}$) equals a vector of 1’s if each agency follows the proposed scheme. For ridge regression, $Y_{s1} = 0$ and $\beta_{s1} = 0$. For secure collaborative learning, the following requirements need to be met to get $\beta_{s1} = 1$ for linear regression or $\beta_{s1} = 0$ for ridge regression.

\begin{enumerate}
  \item For all $j \in \{1, 2, \cdots, K\}$, agency $j$ computes $Y_{s1j} = \sum_{i=1}^{p} x_{ij}$ as the first pseudo response where $x_{ij}$ is the $i$-th column in $X + \Delta$.
  \item Each agency generates random orthogonal matrix, $p \times p$ invertible matrix commutative with $B_0$ and $3 \times 3$ invertible matrix commutative with $C_0$ for encryption.
  \item The cloud returns model results without further perturbation.
  \item All the agencies follow protocol to decrypt received results.
\end{enumerate}

We use the privacy scheme of two-party collaborative learning as an example to illustrate violating any of the 4 requirements can result in failure to get desired $\hat{\beta}_{s1}$. Following the premodeling phase, the released feature matrix is

$$\begin{pmatrix}
A_{21} A_{11}(X_1 + \Delta_1) B_1 B_2 \\
A_{12} A_{22}(X_2 + \Delta_2) B_1 B_2
\end{pmatrix}$$

and the response is

$$\begin{pmatrix}
A_{21} A_{11}[Y_1, Y_{s11}, Y_{s21}] C_1 C_2 \\
A_{12} A_{22}[Y_2, Y_{s12}, Y_{s22}] C_2 C_1
\end{pmatrix}.$$  

If the second requirement is met, the cloud gets $\{B_1^T B_1^T [(X_1 + \Delta_1)^T (X_1 + \Delta_1) + (X_2 + \Delta_2)^T (X_2 + \Delta_2)] B_1 B_2 \}^{-1} B_1^T B_1^T [(X_1 + \Delta_1)^T (Y_1, Y_{s11}, Y_{s21}) C_1 C_2 + (X_2 + \Delta_2)^T (Y_2, Y_{s12}, Y_{s22}) C_2 C_1]$ from linear regression model. Suppose the cloud returns this matrix directly to each agency without perturbing it (the third requirement) and each agency decrypts it following the protocol (the fourth requirement), the decrypted model results are $\{(X_1 + \Delta_1)^T (X_1 + \Delta_1) + (X_2 + \Delta_2)^T (X_2 + \Delta_2)\}^{-1} [(X_1 + \Delta_1)^T (Y_1, Y_{s11}, Y_{s21}) + (X_2 + \Delta_2)^T (Y_2, Y_{s12}, Y_{s22})]$.

Next we show that $Y_{s11} = \sum_{i=1}^{p} x_{i1}$ and $Y_{s12} = \sum_{i=1}^{p} x_{i2}$ ($x_{i1}$ is the $i$-th column of $X_1 + \Delta_1$ and $x_{i2}$ is the $i$-th column of $X_2 + \Delta_2$) guarantee that each agency gets $\beta_{s1} = 1$. The decrypted results can be expressed in the matrix format

$$M \left( \begin{array}{c}
Y_1, Y_{s11}, Y_{s21} \\
Y_2, Y_{s12}, Y_{s22}
\end{array} \right)$$

where $M \triangleq [(X_1^T + \Delta_1^T, X_2^T + \Delta_2^T) (X_1 + \Delta_1) (X_2 + \Delta_2)]^{-1} (X_1^T + \Delta_1^T, X_2^T + \Delta_2^T)$. Because

$$\left( \begin{array}{c}
1 \\
: \\
1
\end{array} \right) = M \left( \begin{array}{c}
X_1 + \Delta_1 \\
X_2 + \Delta_2 \\
1
\end{array} \right),$$

we choose

$$\left( \begin{array}{c}
Y_{s11} \\
Y_{s12}
\end{array} \right) = \left( \begin{array}{c}
X_1 + \Delta_1 \\
X_2 + \Delta_2
\end{array} \right)$$

to get $\beta_{s1} = 1$. In other words, $Y_{s11}$ and $Y_{s12}$ are the sum of columns in $X_1 + \Delta_1$ and $X_2 + \Delta_2$ which is the first requirement. Similarly, let

$$\left( \begin{array}{c}
Y_{s11} \\
Y_{s12}
\end{array} \right) = \left( \begin{array}{c}
X_1 + \Delta_1 \\
X_2 + \Delta_2
\end{array} \right)$$

for ridge regression to get $\beta_{s1} = 0$.

The second and the fourth requirements examine any malicious adversary performed by agencies. The third requirement examines malicious adversary performed by the cloud. So $\beta_{s1}$ tells the truth whether any agency or the cloud deviates from the schemes and acts as malicious adversary.

\subsection*{2) Chosen plaintext attack}

 Agencies participating collaborative learning may generate fake data to perform chosen plaintext attack. As an example, agency 1 performs maliciously by the following three procedures.

$$\begin{array}{|c|c|c|}
\hline
\text{Threat model} & \text{Response} & \text{Feature matrix} \\
\hline
\text{Non-PP} & Y & X \quad (X^T X)^{-1} X^T Y \\
\hline
\text{Semi-honest} & Y & X \quad (X^T X)^{-1} X^T Y \\
\hline
\text{Malicious adversary} & Y_a C & X \quad (X^T X)^{-1} X^T Y \\
\hline
\end{array}$$

Fig. 3: Linear regression models resilient to different threat models. Non-PP: non-privacy preserving model. $Y_a = [Y, Y_{s1}, Y_{s2}]$. 

\begin{table}[h]

\begin{tabular}{|c|c|c|c|c|}
\hline
\text{Threat model} & \text{Response} & \text{Feature matrix} & \text{\hat{\beta}} & \text{\hat{\beta} (after post-modeling phase)} \\
\hline
\text{Non-PP} & Y & X & (X^T X)^{-1} X^T Y & - \\
\hline
\text{Semi-honest} & Y & A(X + \Delta) B & B^{-1}((X + \Delta)^T (X + \Delta))^{-1} (X + \Delta)^T Y & ((X + \Delta)^T (X + \Delta))^{-1} (X + \Delta)^T Y \\
\hline
\text{Malicious adversary} & Y_a C & A(X + \Delta) B & B^{-1}((X + \Delta)^T (X + \Delta))^{-1} (X + \Delta)^T Y_a C & ((X + \Delta)^T (X + \Delta))^{-1} (X + \Delta)^T Y_a \\
\hline
\end{tabular}
\end{table}
1) First, agency 1 sends $X_1^* = A_{11}(X_1 + \Delta_1)B_1$ to agency 2.

2) Second, agency 2 encrypts $X_1^*$ as $X_{1\text{new}} = A_{21}X_1^*B_2$ where $A_{21}$ is random orthogonal matrix and $B_2 = \sum_{j=1}^p b_j^0B_0^j$ ($b_j^0$ is random coefficient). Then agency 2 releases the encrypted data.

3) Because only invertible encryption matrix basis $B_0$ is released to agencies, agency 1 generates random orthogonal matrix $A_1^*$ and $B_2 = \sum_{j=1}^p b_j^0B_0^j$ ($b_j^0$ is random parameter) and then uses equation $A_{21}X_1^*B_2 = A_1^*X_1^*B_2$ to recover $B_2$.

Because $\hat{B}_2 = \sum_{j=1}^p b_j^0B_0^j$, $\hat{B}_2$ can also be written as follows.

$$\hat{B}_2 = (B_0 B_0^2 B_0^3 \cdots B_0^p) \begin{bmatrix} b_1^0 I \\ b_2^0 I \\ \vdots \\ b_p^0 I \end{bmatrix}$$

where $I$ is identity matrix. So $X_{1\text{new}} = A_1^*X_1^*\hat{B}_2$ can be written as

$$X_{1\text{new}} = A_1^*X_1^*(B_0 B_0^2 B_0^3 \cdots B_0^p) \begin{bmatrix} b_1^0 I \\ b_2^0 I \\ \vdots \\ b_p^0 I \end{bmatrix}$$

Let $U \triangleq \begin{bmatrix} b_1^0 I \\ b_2^0 I \\ \vdots \\ b_p^0 I \end{bmatrix}$. The $p$ unknown parameters are all included in each column of $U$. So the above equation can be broken down into $p$ sub-equations. With the $j$-th column of $U$ being the unknown vector $(u_j)$ and the $j$-th column of $X_{1\text{new}}$ being $w_j$, we have

$$w_j = A_1^*X_1^*(B_0 B_0^2 B_0^3 \cdots B_0^p)u_j, \quad j = 1, \cdots, p.$$

Let $R \triangleq A_1^*X_1^*(B_0 B_0^2 B_0^3 \cdots B_0^p)$ and then $w_j = R_{\cdot j}u_j$. The dimension of $R$ is $n \times p^2$. To solve $u_j$, we first release the restriction of $u_j$ and do not restrict to $p$ unknown parameters. The solution is related to the rank of $R$. More specifically,

1) if $\text{rank}(R) < \text{rank}([R, w_j])$, there is no solution for $u_j$;

2) if $\text{rank}(R) = \text{rank}([R, w_j]) = p^2$, there is a unique solution for $u_j$;

3) if $\text{rank}(R) = \text{rank}([R, w_j]) < p^2$, there are infinite solutions for $u_j$.

Because $\text{rank}(R) \leq \min\{\text{rank}(A_1^*X_1^*), \text{rank}([B_0 B_0^2 B_0^3 \cdots B_0^p])\}$, we have $\text{rank}(R) \leq \min\{n, p\} < p^2$. So it is impossible to have unique solution for $u_j$. If $\text{rank}(X_1^*) = \min\{n, p\}$, we have $\text{rank}(R) = \min\{n, p\}$ because $\text{rank}(B_0) = p$ and $\text{rank}(A_1^*) = n$. Then the solution has a direct relation with the dimension of $X_1^*$ as listed below.

1) For $n \geq p+1$, $\text{rank}(R) = p$ and $\text{rank}([R, w_j]) = p+1$.

So there is no solution for $u_j$;

2) For $n \leq p$, $\text{rank}(R) = \text{rank}([R, w_j]) = n < p^2$. So there are infinite solutions for $u_j$.

This applies for all the $p$ sub-equations ($j = 1, \cdots, p$). Because orthogonal matrix for encryption is randomly generated by each agency, the true matrix $B_2$ is not a solution, i.e., $X_{1\text{new}} \neq A_1^*X_1^*B_2$. A toy example is given in Appendix A. Each of the $p$ sub-equations derives different solutions of $b_j$ ($j = 1, \cdots, p$) or there is no solution. Moreover, different $A_1^*$ in the equation gets different solutions. So the proposed encryption scheme is resilient to chosen plaintext attack by malicious agencies participating collaborative learning.

B. Known plaintext attack

With both the encrypted data and partial original data released, known plaintext attack is an effective approach to recover sensitive information from data encrypted by multiplicative perturbation, e.g., $AX$ and $XB$. For the proposed encryption method, we show that encryption matrices $A$ and $B$ protect against known plaintext attack.

Suppose the adversary knows partial data (denoted as $X_{11}$) in the sensitive data. The first scenario (I) is

$$X = \begin{pmatrix} X_{11} \\ X_{22} \end{pmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

and the second scenario (II) is

$$X = (X_{11}, X_{22}) = \begin{pmatrix} x_{11} & \cdots & x_{1p1} & x_{1(p1+1)} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p1} & x_{2(p1+1)} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{n1} & \cdots & x_{np} \end{pmatrix}$$

where the elements highlighted in pink color denote data disclosed to the adversary and elements highlighted in green color denote private data.

I. For $n \times p$ dimensional $X = \begin{pmatrix} X_{11} \\ X_{22} \end{pmatrix}$, the released data is $X^* = A(X + \Delta) = A \begin{pmatrix} X_{11} + \Delta_{11} \\ X_{22} + \Delta_{22} \end{pmatrix} = \begin{pmatrix} X_{11}^* \\ X_{22}^* \end{pmatrix}$.

The adversary has equation $\hat{A}^TX^* = (X_{11} + \Delta_{11})/X_{22} + \Delta_{22}$ where matrices with $\hat{\cdot}$ denote recovered matrices. Because $A$ and $\hat{A}^T$ are orthogonal matrices, the adversary has equation $X^T X^* = X^T \hat{A} \hat{A}^T X^* = (X_{11} + \Delta_{11}, X_{22} + \Delta_{22}) (X_{11} + \Delta_{11})/X_{22} + \Delta_{22}$.

The equation can be simplified as

$$X^T X^* = X_{11}^T X_{11} + X_{22}^T X_{22}$$
by assuming $\Delta_{11} = \Delta_{22} = 0$. Any orthogonal transformation of $X_{22}$ satisfying this equation can be a recovered $X_{22}$ by the adversary. Given
\[ X^T X = (X_{11} + \Delta_{11})(X_{11} + \Delta_{11}) + (X_{22} + \Delta_{22})(X_{22} + \Delta_{22}), \]
we have
\[ \hat{X}_{22}^T \hat{X}_{22} - X_{22}^T X_{22} = (X_{11} + \Delta_{11})(X_{11} + \Delta_{11}) - X_{11}^T X_{11} + \hat{X}_{22}^T \Delta_{22} + \Delta_{22}^T X_{22} + \Delta_{22}^T \Delta_{22}. \]

Consider a simplified scenario assuming that the disclosed data $X_{11}$ and private data $X_{22}$ are not mixed together, i.e., they are encrypted by separate orthogonal matrices. Then we have
\[ X^T X^* - X_{11}^T X_{11} = X^T X^* - X_{11}^T X_{11} = X_{22}^T X_{22}. \]

After getting $X_{22}^T X_{22}$, the adversary recovers $\hat{X}_{22}$ based on $X_{22}^T X_{22} = X_{22}^T X_{22}^*$. Since
\[ X_{22}^T X_{22}^* = (X_{22} + \Delta_{22})^T (X_{22} + \Delta_{22}), \]
any orthogonal transformation of $X_{22} + \Delta_{22}$ can be a possible recovered $\hat{X}_{22}$. So
\[ \hat{X}_{22}^T \hat{X}_{22} - X_{22}^T X_{22} = X_{22}^T \Delta_{22} + \Delta_{22}^T X_{22} + \Delta_{22}^T \Delta_{22}. \]

II. For $n \times p$ dimensional $X = (X_{11}, X_{22})$, the released data is $X^* = A(X + \Delta) = A(X_{11} + \Delta_{11}, X_{22} + \Delta_{22}) = (X_{11}, X_{22})$. Assume $X^*$ and $X_{11}$ are disclosed and the adversary tries to recover $X_{22}$ by known plaintext attack. The adversary has equation $X_{11} = A(X_{11} + \Delta_{11})$ where $A$ and $\Delta_{11}$ are the recovered encryption matrices by the adversary. Let $\hat{X}_{11}$ be 0 and the simplified equation is $X_{11} = A X_{11}$. Practically the adversary only knows limited information of the sensitive data $X$, i.e., $X_{11}$ has small number of columns and is not able to recover encryption matrix $A$. Consider the extreme case where $X_{11}$ contains at least $n$ columns and thus $X_{11} + \Delta_{11}$ is invertible. The adversary performs the following two computation steps to recover $\hat{X}_{22}$.

1. $A(X_{11} + \Delta_{11}) = A X_{11} \Rightarrow A^T A = X_{11} (X_{11} + \Delta_{11})^{-1}$;
2. $X_{22} = A^T X_{22}^* = A^T A (X_{22} + \Delta_{22}) = X_{11} (X_{11} + \Delta_{11})^{-1} (X_{22} + \Delta_{22}).$

The above discussion of known plaintext attack is based on the encryption function $f(X) = A(X + \Delta)$. The recovered data derived above is further perturbed by $B$. We already prove that encryption matrix $B$ is impossible to be recovered by malicious adversary (Section VI-A2). For $\Delta = 0$, we show that the proposed scheme is still secure with the encryption matrices $A$ and $B$.

The contribution of invertible matrix $B$ Because $B$ can not be recovered and separated from $X_B$, we replace $X$ with $X_B$ in recovered data derived above.

- In scenario I, the adversary has equation $X^T X^* = X_{11}^T X_{11} + \hat{X}_{22}^T \hat{X}_{22}$ where $X^T X^* = B^T [(X_{11}^T + \Delta_{11}^T)(X_{11} + \Delta_{11}) + (X_{22}^T + \Delta_{22}^T)(X_{22} + \Delta_{22})]B$. So $X_{22}^T X_{22} - X_{22}^T X_{22}$ has lower bound as follows.
\[ \hat{X}_{22}^T \hat{X}_{22} - X_{22}^T X_{22} = X^T X^* - X_{11}^T X_{11} - X_{22}^T X_{22} \]
\[ > B^T (X_{22} + \Delta_{22}^T)(X_{22} + \Delta_{22})B - X_{22}^T X_{22}. \]

Consider a simple case without additive noise (i.e., $\Delta_{11} = \Delta_{22} = 0$). The difference between $X_{22}^T X_{22}$ and $X_{22}^T X_{22}$ is
\[ B^T (X_{11}^T X_{11} + X_{22}^T X_{22})B - X_{11}^T X_{11} - X_{22}^T X_{22} \]
with lower bound $B^T X_{22}^T X_{22} B - X_{22}^T X_{22}$.

- In scenario II, the columns of $X_{11}$ and $X_{22}$ are encrypted and mixed together by invertible matrix $B$. Because the adversary can not recover $B$, the adversary gets $\hat{X}_{22} = X_{11}(Z_{11}^{-1})X_{22}$ where $(Z_{11}, Z_{22}) = (X_{11} + \Delta_{11}, X_{22} + \Delta_{22}) B$. Consider a simple case where $X_{11}$ and $X_{22}$ are encrypted by separate invertible matrices $B_1$ and $B_2$ (i.e., their columns are not mixed together). The recovered $\hat{X}_{22}$ can be simplified as $\hat{X}_{22} = X_{11}(X_{11} + \Delta_{11})^{-1} X_{22} + \Delta_{22} B_2$. Then $\hat{X}_{22} = X_{11} B_1^{-1} X_{11}^{-1} X_{22} B_2$ for $\Delta_{11} = \Delta_{22} = 0$. With the perturbation of the random invertible matrices $B_1$ and $B_2$, $\hat{X}_{22}$ deviates from $X_{22}$ without additive noise.

For random invertible matrix with element following normal distribution $N(0, \sigma_B)$, we show that each element in $B^T X_{22}^T X_{22} B$ has an upper bound.

The $(i, j)$-th element in $B^T X_{22}^T X_{22} B$ (i.e., element in the $i$-th row and the $j$-th column) is the product of the $i$-th row in $B^T X_{22}^T X_{22}$ and the $j$-th column in $X_{22} B$. The $j$-th column in $X_{22} B$ can be expressed as
\[ \begin{pmatrix} x_{1j}^* \\ x_{2j}^* \\ \vdots \\ x_{nj}^* \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{pmatrix} \]

where $(b_{1j}, b_{2j}, \ldots, b_{nj})^T$ is the $j$-th column in $B$. Because $b_{kj} = \sum_{i=1}^{\sigma_B} \sigma_B^2$ follows normal distribution $N(0, \sigma_B^2)$ for $c = 1, \ldots, n$. Since the $i$-th row in $B^T X_{22}^T X_{22} B$ is the same as the $i$-th column in $X_{22} B$, the $(i, j)$-th element in $B^T X_{22}^T X_{22} B$ equals $\sum_{t=1}^{\sigma_B} x_{ti}^* x_{tj}^*$. For each $t \in \{1, \ldots, n\}$, $x_{ti}^* x_{tj}^*$ follows Gamma distribution $\Gamma(1/2, 2\sigma_B^2 \sum_{k=1}^{\sigma_B} x_{tk}^2)$. Given $n$ and $X$, $\sum_{k=1}^{\sigma_B} x_{tk}^2$ is fixed. For $\sigma_B \rightarrow 0$, $x_{ti}^* x_{tj}^* \rightarrow 0$. So $\sum_{t=1}^{\sigma_B} x_{ti}^* x_{tj}^* \rightarrow 0$.

The encryption method $AXB$ is sufficient to protect against known plaintext attack without adding additive noise. So we set $\Delta = 0$ to ensure high data utility.

C. Local differential privacy

To achieve local differential privacy (LDP), invertible matrix $B$ is generated randomly with each element following normal distribution $N(0, \sigma_B^2)$.

As proved in [43], [44], Johnson-Lindenstrauss (JL) transformation preserves differential privacy (DP). Given $n \times d$ matrix $X$ and $r \times n$ encryption matrix $R$ with each entry following Gaussian distribution $N(0, 1)$, $RX$ preserves $(\epsilon, \delta)$-DP for a specified $r$. Here we prove $XB$ achieves LDP if
entries in $B$ follow Gaussian distribution. The approaches in [43], [44] encrypt sensitive data but also need to ensure that encrypted data has acceptable data utility. In our method, the data utility is guaranteed by the property of the encryption matrix. The encryption matrix multiplication is commutative and the encryption effect can be eliminated by the post-modeling phase. Since the encryption can be eliminated in our method, the perturbation of $B$ guarantees LDP and also does not influence data utility.

**Theorem 1.** Encryption function $f(X) = XB_0$ achieves LDP if entries in encryption matrix $B_0$ follow Gaussian distribution $N(0, \sigma_{B_0}^2)$.

**Proof.** Consider any two samples, $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \ldots, x_p^{(1)})$ and $x^{(2)} = (x_1^{(2)}, x_2^{(2)}, \ldots, x_p^{(2)})$, that randomly chosen from all the possible inputs in $X$.

$$x^{(1)}B_0 = (x_1^{(1)}b_1, x_2^{(1)}b_2, \ldots, x_p^{(1)}b_p) = (x_1^{(1)}, x_2^{(1)}, \ldots, x_p^{(1)}),$$

$$x^{(2)}B_0 = (x_1^{(2)}b_1, x_2^{(2)}b_2, \ldots, x_p^{(2)}b_p) = (x_1^{(2)}, x_2^{(2)}, \ldots, x_p^{(2)}).$$

Each element in $x^{(1)}B_0$ or $x^{(2)}B_0$ is the linear combination of $B_0$ as shown in the matrix multiplication above. Because each element in $B_0$ follows normal distribution $N(0, \sigma_{B_0}^2)$, each element in $x^{(1)}B_0$ also follows normal distribution. Figure 4 shows that the values of $xB_0$ is close to 0 as $\sigma_{B_0} \to 0$ for any row $x$ in $X$. Specifically, each element in $x^{(1)}B_0$ follows $N(0, ||x^{(1)}||_2^2\sigma_{B_0}^2)$ and each element in $x^{(2)}B_0$ follows $N(0, ||x^{(2)}||_2^2\sigma_{B_0}^2)$. When $\sigma_{B_0} \to 0$, these two distributions are close to each other. We use the first element $x_1^{(1)}$ in $x^{(1)}B_0$ and the first element $x_1^{(2)}$ in $x^{(2)}B_0$ as an example to show LDP achieved by encryption matrix $B_0$.

$$\frac{P(x_1^{(1)} \in (-t,t))}{P(x_1^{(2)} \in (-t,t))} = \text{erf}(t/\sqrt{2||x^{(1)}||_2^2\sigma_{B_0}^2})$$

Figure 5 shows $\frac{P(x_1^{(1)} \in (-t,t))}{P(x_1^{(2)} \in (-t,t))} \to 1$ when $\sigma_{B_0} \to 0$ for two examples. In other words, the encryption method achieves local differential privacy.

**Theorem 2.** Given the commutative encryption matrix $B = \sum_{j=1}^{p} b_jB_0^j$, $f(X) = XB$ achieves LDP.

**Proof.** Suppose data is encrypted in the form of $X(\sum_{j=1}^{p} b_jB_0^j)$. The encrypted data can be expressed as $b_1XB_0 + b_2XB_0^2 + \cdots + b_pXB_0^p = XB_0(b_1 + b_2B_0 + \cdots + b_pB_0^{p-1})$. It is encrypted by two encryption functions $f_1(X) = XB_0$ and $f_2(X^*) = X^*(b_1 + b_2B_0 + \cdots + b_pB_0^{p-1})$. $f_1(X) = XB_0$ achieves LDP if entries in $B_0$ follow Gaussian distribution (Theorem 1). According to closure under postprocessing property [45], $f(X) = f_2(f_1(X))$ achieves local differential privacy.

Similarly, $AXB$ achieves LDP based on closure under postprocessing property. With the encryption matrix $B$, the cloud server gets encrypted model estimate $B^{-1}\hat{\beta}C$ (Section V-B) where $B$ and $C$ are encryption matrices. The invertible encryption matrices are decrypted in the post-modeling phase to get accurate model estimate and prediction accuracy remains the same as non-secure model. Data $X$ remains encrypted and secure all the time.

**D. Collusion attack**

The proposed encryption scheme is resilient to malicious adversary which compromises all but one agency. In this study, any released data is accessible to all agencies and the proposed encryption scheme is resilient to chosen plaintext attack and known plaintext attack. The resilience of chosen plaintext attack shows that encryption matrix $B$ can not be recovered. Any collusion among agencies is not able to provide extra information for the attack. Because the data is
The experiments on real data show that our privacy preserving scheme is efficient (Figure 6) and requires less computation time compared to existing secure schemes. It takes 46 seconds to analyze YearPredictionMSD data if 100 agencies participate in the collaborative learning and each agency has around 5,000 samples. A maliciously secure linear model, Helen, takes around 1.5 hours to finish model computation for four collaborating agencies with each holding 1,000 samples [6]. Another maliciously secure linear regression scheme, GuardLR [25], takes around 218 seconds to build two-party secure linear model using Thyroid disease data. It takes 30 seconds to perform prediction on Diabetes data in a differentially private scheme [48]. Credit card data was used to evaluate the secure prediction scheme proposed in [49]. However, the model takes more than 1 hour assuming the entire sample is partitioned into 16 disjoint subsets. For Thyroid disease data, Diabetes data, and Credit card data, our scheme takes less than 10 seconds to conduct secure linear regression. Our privacy preserving scheme is in general more efficient than other secure schemes proposed previously.

B. Privacy protection

We choose the first 100 samples in each dataset and encrypts data with orthogonal matrix $A$ and invertible matrix $B$ with zero mean and SD=0.001. We show that the proposed encryption scheme is resilient to known plaintext attack (scenario I and II).

Because the number of features (i.e., $p$) is small in each dataset, it is possible for the adversary to collect all the $p$ features of some samples and then perform known plaintext attack under scenario I. Suppose $X_{11}$ contains the first $p$ samples and $p$ features, $X_{22}$ contains the remaining $100-p$ samples with $p$ features. Figure 7 shows that the recovered matrix $X_{22}^TX_{22}$ shrinks to 0 as $\sigma_B \to 0$ for the simplified scenario I discussed in Section VI-B. Because the encryption matrix $B$ can be recovered in the post-modeling phase, we choose $\sigma_B \to 0$ to protect against known plaintext attack and also to achieve local differential privacy.

To simulate known plaintext attack under scenario II, we assume the first $p-5$ features are disclosed, i.e., $X_{11}$ contains $p-5$ features with $p-5$ samples. $X_{22}$ contains the other 5 features for these $p-5$ samples. Figure 8 shows the difference between recovered data $\hat{X}_{22}$ and original data $X_{22}$ for a randomly selected feature among 5 non-disclosed features. Different from scenario I, $\sigma_B$ does not affect the amplitude of the recovered data. Because the invertible matrix is randomly

VII. PERFORMANCE EVALUATION

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.

We perform experiments using four datasets from the UCI repository [46]. All the experiments are performed in Matlab on University of Florida Hipergator 3.0 with 1 CPU and 4 RAMs.

YearPredictionMSD: This dataset contains 515,345 songs with 90 features. The goal is to predict a song’s published year using these 90 features. The prediction of linear regression model is rounded as the predicted year.

Thyroid disease dataset: Thyroid contains 7200 samples and 21 features. The response is normal (not hypothyroid) or abnormal.

Diabetes 130-US hospitals: This dataset was developed to identify factors related to readmission for patients with diabetes. The features were preprocessed following the procedure in [47]. A total of 69,977 samples and 42 features were used to build model predicting if the patient was readmitted within 30 days of discharge (yes/no).

Default of credit card clients Data Set: There are 30,000 samples and 23 features in the Taiwan credit dataset. The goal is to predict default payment (yes/no).

We assume the number of samples in each dataset is equally split into $K$ agencies while each subset contains all the features. The computation cost of the pre-modeling phase is increased when the number of agencies $K$ is increased. The privacy level and data utility are not related to $K$ since the total number of samples is fixed.

A. Efficiency

The experiments on real data show that our privacy preserving scheme is efficient (Figure 6) and requires less computation time compared to existing secure schemes. It takes 46 seconds to analyze YearPredictionMSD data if 100 agencies participate in the collaborative learning and each agency has around 5,000 samples. A maliciously secure linear model, Helen, takes around 1.5 hours to finish model computation for four collaborating agencies with each holding 1,000 samples [6]. Another maliciously secure linear regression scheme, GuardLR [25], takes around 218 seconds to build two-party secure linear model using Thyroid disease data. It takes 30 seconds to perform prediction on Diabetes data in a differentially private scheme [48]. Credit card data was used to evaluate the secure prediction scheme proposed in [49]. However, the model takes more than 1 hour assuming the entire sample is partitioned into 16 disjoint subsets. For Thyroid disease
generated, the perturbation is sufficient and the encrypted data is resilient to the attack under scenario II.

C. Prediction accuracy

We evaluate model performance using 10-fold cross validation with 10 iterations. In the cross validation, samples are randomly split into 10 folds with the equal size with 9 folds for training and 1 fold for testing. We use mean square error (MSE) to evaluate the prediction performance for continuous response and the area under the receiver operating characteristic curve (AUC) to evaluate accuracy in classification problems. The average MSE and AUC of 10 iterations for four datasets are shown in Figure 9. The variation of MSE/AUC across different $\sigma_B$ is caused by different sample selection in the cross validation. Because model estimate is decrypted in the post-modeling phase, our scheme (regardless of the choice of $\sigma_B$) gets the same model result compared to non-privacy preserving linear regression using same samples for training and testing.

The proposed privacy preserving linear regression is efficient and has competitive accuracy when analyzing binary response in some cases. As discussed in previous studies, it is not restricted to use logistic regression with a binary response and there are compelling substantive arguments for preferring linear regression in many cases [50], [51]. Compared to existing differentially private logistic regression [48], [49], our privacy preserving linear regression achieves similar accuracy levels and requires less computation cost. Moreover, we provide higher level of privacy protection which is resilient to malicious adversary.

VIII. CONCLUSION

In this paper, we propose efficient privacy preserving schemes for collaborative linear model when data is distributed among different agencies. The proposed scheme is against malicious adversaries, including chosen plaintext attack, known plaintext attack, and collusion attack. The proposed scheme also satisfies local differential privacy. It is efficient to conduct cross validation without additional communication cost. The experimental analysis shows that our scheme is more effi-
cienct than existing secure linear regression techniques against malicious adversary. Our privacy preserving scheme requires data encrypted by each agency participated in the collaborative learning which increases communication cost. As the recent development of data compression technology [22], data can be compressed before transforming among agencies to reduce communication cost. In the future, we are interested in extending the privacy preserving schemes to other statistical models.

Appendix A

Toy Example of Chosen Plaintext Attack

We give an example to show that the proposed encryption scheme is resilient to chosen plaintext attack. We set \( n = p = 3 \) and follow the procedure described in Section VI-A2.

1) Generate a random invertible matrix \( B_0 \) with each element following normal distribution \( N(0, 1) \).

2) Agency 1 generates a \( 3 \times 3 \) random matrix \( X_1^* \) with each element following normal distribution \( N(1, 1) \) and sends \( X_1^* \) to agency 2.

3) Agency 2 encrypts \( X_1^* \) as \( X_{1new}^* = A_2 X_1^* B_2 \) where \( A_2 \) is random orthogonal matrix and \( B_2 = 8 B_0 + 0.3 B_0^* - 2 B_0^3 \).

4) Agency 1 sets identity matrix as the orthogonal encryption matrix (i.e., \( A_1^* = I \)) and \( B_1 = b_1^* B_0 + b_2^* B_2^3 + b_3^* B_0^3 \). Equation \( X_{1new}^* = A_1^* X_1^* B_2 = X_1^* B_2 \) is used to recover \( B_2 \).

Example: We first list the matrices generated for the chosen plaintext attack. \( B_0 = \begin{pmatrix} -0.626 & 1.595 & 0.487 \\ 0.184 & 0.330 & 0.738 \\ -0.816 & -0.820 & 0.576 \end{pmatrix} \). \( X_1^* = \begin{pmatrix} 0.695 & 0.379 & 0.955 \\ 2.512 & -1.215 & 0.984 \\ 1.390 & 2.125 & 1.944 \end{pmatrix} \).

\[ X_{1new}^* = \begin{pmatrix} 7.517 & -5.452 & -0.865 \\ 11.13 & -16.98 & -2.979 \\ 17.12 & -23.77 & -38.04 \end{pmatrix} \]

\[ X_1^* \begin{pmatrix} b_1^* \\ b_2^* \\ b_3^* \end{pmatrix} \left( \begin{pmatrix} b_1^* \\ b_2^* \\ b_3^* \end{pmatrix} \right) = X_1^* \begin{pmatrix} b_0^* \\ b_2^* \\ b_3^* \end{pmatrix} \]

It can be broken down into 3 sub-equations. Let \( R = X_1^* \begin{pmatrix} B_0 \\ B_0^* \\ B_0^3 \end{pmatrix} \). The first equation is \( w_1^* = R w_1 \) where \( w_1 \) is the first column of \( X_{1new}^* \) and \( w_1 \) is the first column of \( \begin{pmatrix} b_1^* \\ b_2^* \\ b_3^* \end{pmatrix} \). The reduced row echelon form of \( (R, w_1) \) is

\[
\begin{pmatrix}
1 & 0 & 0 & -0.63 & 1.60 & 0.49 & -0.87 & 1.15 & -3.63 \\
0 & 1 & 0 & 0.18 & 0.33 & 0.74 & -0.67 & -0.20 & 0.76 & -0.36 \\
0 & 0 & 1 & -0.84 & -0.82 & 0.58 & -0.11 & 2.08 & -0.68 & 2.96
\end{pmatrix}
\]

So the sub-equation has infinite solutions \( w_1 = \begin{pmatrix} -0.67 \\ 0.18 \\ -0.84 \end{pmatrix} v_4 + \begin{pmatrix} 1.60 \\ 0.33 \\ 0 \end{pmatrix} v_5 + \begin{pmatrix} 0.49 \\ 0.74 \\ 0 \end{pmatrix} v_6 + \begin{pmatrix} -0.87 \\ -0.67 \\ 0 \end{pmatrix} v_7 + \begin{pmatrix} 1.15 \\ 0.76 \\ 0 \end{pmatrix} v_8 + \begin{pmatrix} -3.63 \\ -0.36 \\ 0 \end{pmatrix} v_9 \)

where \( v_i \) (\( i = 4, \ldots, 9 \)) can be any value. Then we add the restriction back to \( w_1 \) (i.e., the second, third, 5th, 6th, 8th, 9th elements equal 0). So \( v_5 = v_6 = v_8 = v_9 = 0 \), \( v_4 = b_1^* \), \( v_7 = b_3^* \), \( v_4 = 0.28 \), \( v_7 = -0.67 \), \( v_1 = -3.63 \), \( v_2 = -0.84 \), \( v_6 = -0.47 \), \( v_1 = 2.96 \). Agency 1 gets \((b_1, b_2, b_3) = (-5.71, 3.47, 0.40)\). Similarly, the second sub-equation gets \((b_1, b_2, b_3) = (-11.0, -1.65, 4.15)\) and the third sub-equation gets \((b_1, b_2, b_3) = (-3.45, -4.90, 5.24)\).

The example shows that different sub-equations get different solutions for unknown parameters of \( B_2 \) and none of the solutions are close to the true values \((b_1, b_2, b_3) = (8, 0.3, -2)\).

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 In 2013 IEEE Symposium on Security and Privacy, 2013, pp. 334–348.

[3] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, 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. Giacomi, 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.

[7] Y. Hu, A. Shyngyskova, D. Tran, N. Eriksson, J. Tung, and D. Hinds, “GWAS of 89,283 individuals identifies genetic variants associated with self-reporting of being a morning person,” Nature Communications, vol. 7, p. 10448, 2016.

[8] K. Valaskova, T. Kliestik, L. Svabova, and P. Adamko, “Financial risk measurement and prediction modelling for sustainable development of business entities using regression analysis,” Sustainability, vol. 10, no. 7, 2018.

[9] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” Found. Trends Theor. Comput. Sci., vol. 9, no. 3–4, p. 211–407, Aug. 2014.

[10] C. Aggarwal and P. Yu, “A condensation approach to privacy preserving data mining,” in Proceedings of International Conference on Extending Database Technology, Heraklion,Crete,Greece, 2004.

[11] K. Chen and L. Liu, “Geometric data perturbation for privacy preserving outsourced data mining,” Knowledge and Information Systems, vol. 29, no. 3, pp. 657–695, 2011.

[12] K. Liu, H. Kargupta, and J. Ryan, “Random-projection-based multiplicative data perturbation for privacy preserving distributed data mining,” IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 1, pp. 92–106, 2006.

[13] W. Du, Y. S. Han, and S. Chen, “Privacy-preserving multivariate statistical analysis: linear regression and classification.” in Proceedings of the 4th SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 2004.

[14] A. Karr, X. Lin, A. Sanil, and J. Reiter, “Privacy-preserving analysis of vertically partitioned data using secure matrix products.” Journal of Official Statistics, vol. 25, no. 1, pp. 125–138, 2009.
[15] S. Wu, S. Chen, D. Burr, and L. Zhang, “A new data collection technique for preserving privacy.” Journal of Privacy and Confidentiality, vol. 7, no. 3, p. 326, 2017.

[16] F. Chen, T. Xiang, X. Lei, and J. Chen, “Highly efficient linear regression outsourcing to a cloud.” IEEE Transactions on Cloud Computing, vol. 2, no. 4, pp. 499–508, 2014.

[17] Y. Zhang, X. Xiao, L. Yang, Y. Xiang, and S. Zhong, “Secure and efficient outsourcing of PCA-based face recognition.” IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1683–1695, 2020.

[18] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, “New algorithms for secure outsourcing of large-scale systems of linear equations.” IEEE Transactions on Information Forensics and Security, vol. 10, no. 1, pp. 69–78, 2015.

[19] S. Zhang, C. Tian, H. Zhang, J. Yu, and F. Li, “Practical and secure outsourcing algorithms of matrix operations based on a novel matrix encryption method.” IEEE Access, vol. 7, pp. 53823–53838, 2019.

[20] L. Zhao and L. Chen, “Sparse matrix masking-based non-interactive verifiable (outsourced) computation, revisited.” IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 6, pp. 1188–1206, 2020.

[21] M. Dzwonkowski and R. Rykaczewski, “Secure quaternion feistel cipher for dicom images,” IEEE Transactions on Image Processing, vol. 28, no. 1, pp. 371–380, 2019.

[22] Z. Cao, L. Liu, and O. Markowitch, “Comment on ‘highly efficient linear regression outsourcing to a cloud.’” IEEE Transactions on Cloud Computing, vol. 7, no. 3, pp. 893–893, 2019.

[23] R. Hall, S. Fienberg, and Y. Nardi, “Secure multiple linear regression based on homomorphic encryption.” Journal of Official Statistics, vol. 27, no. 4, pp. 669–691, 2011.

[24] M. Cock, R. Dowseley, A. Nascimento, and S. Newman, “Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data.” in Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, Denver, Colorado, USA, October 2015.

[25] Z. Ma, J. Ma, Y. Xiao, L. Liu, K.-K. R. Choo, Y. Gao, and R. H. Deng, “Verifiable data mining against malicious adversaries in industrial internet of things.” IEEE Transactions on Industrial Informatics, vol. 18, no. 2, pp. 953–964, 2022.

[26] A. Sanil, A. Karr, X. Lin, and J. Reiter, “Privacy preserving regression modelling via distributed computation.” in 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Seattle, WA, USA, August 2004.

[27] A. Karr, X. Lin, A. Sanil, and J. Reiter, “Secure regression on distributed databases.” Journal of Computational and Graphical Statistics, vol. 14, no. 2, pp. 263–279, 2005.

[28] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, “CHET: An optimizing compiler for fully-homomorphic neural-network inferencing,” in Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, ser. PLDI 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 142–156.

[29] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, “Functional mechanism: Regression analysis under differential privacy.” Proc. VLDB Endow., vol. 5, no. 11, pp. 1364–1375, 2012.

[30] A. Nikolov, K. Talwar, and L. Zhang, “The geometry of differential privacy: The sparse and approximate cases,” in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing. New York, NY, USA: Association for Computing Machinery, 2013, p. 351–360.

[31] T. Nguyen, X. Xiao, Y. Yang, S. Hui, H. Shin, and J. Shin, “Collecting and analyzing data from smart device users with local differential privacy.” CoRR, 06 2016.

[32] O. Sheffer, “Differentially private ordinary least squares,” in Proceedings of the 34th International Conference on Machine Learning - Volume 70, 2017, p. 3105–3114.

[33] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, “Collecting and analyzing multidimensional data with local differential privacy.” in 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019, pp. 638–649.

[34] D. Alabi, A. McMillan, J. Sarathy, A. Smith, and S. Vadhan, “Differentially private simple linear regression,” Proceedings on Privacy Enhancing Technologies, vol. 2022, pp. 184–204, 04 2022.

[35] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” Technometrics, vol. 12, no. 1, pp. 55–67, 1970.

[36] S. Li, C. Li, G. Chen, N. G. Bourbakis, and K.-T. Lo, “A general quantitative cryptanalysis of permutation-only multimedia ciphers against plaintext attacks,” Signal Processing: Image Communication, vol. 23, no. 3, pp. 212–223, 2008.

[37] K. Liu, C. Giannella, and H. Kargupta, “An attacker’s view of distance preserving maps for privacy preserving data mining,” in Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 2006, pp. 297–308.

[38] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, “Manipulating machine learning: Poisoning attacks and countermeasures for regression learning,” in 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 19–35.

[39] D. Agrawal and C. C. Aggarwal, “On the design and quantification of privacy preserving data mining algorithms,” in Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ser. PODS ’01. New York, NY, USA: Association for Computing Machinery, 2001, p. 247–255.

[40] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, “On the privacy preserving properties of random data perturbation techniques,” in Third IEEE International Conference on Data Mining, 2003, pp. 99–106.

[41] Z. Huang, W. Du, and B. Chen, “Deriving private information from randomized data,” in Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’05. New York, NY, USA: Association for Computing Machinery, 2005, p. 37–48.

[42] R. H. Keshavan, A. Montanari, and S. Oh, “Matrix completion from noisy entries,” J. Mach. Learn. Res., vol. 11, pp. 2057–2078, 2010.

[43] J. Blocki, A. Blum, A. Datta, and O. Sheffet, “The johnson-lindenstrauss transform itself preserves differential privacy,” in 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, 2012, pp. 410–419.

[44] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren, “Dppro: Differentially private high-dimensional data release via random projection,” IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 3081–3093, 2017.

[45] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” Journal of Privacy and Confidentiality, vol. 7, no. 3, p. 17–51, May 2017.

[46] “UCI machine learning repository.” [Online]. Available: [http://archive.ics.uci.edu/ml]