Privacy Preserving Linear Regression on Distributed Databases

Fida K. Dankar*†
* Sidra Medical and Research Center, Doha, Qatar
fdankar@sidra.org
† IBM, Toronto, Canada
fidamark@ca.ibm.com

Abstract. Studies that combine data from multiple sources can tremendously improve the outcome of the statistical analysis. However, combining data from these various sources for analysis poses privacy risks. A number of protocols have been proposed in the literature to address the privacy concerns; however they do not fully deliver on either privacy or complexity. In this paper, we present a (theoretical) privacy preserving linear regression model for the analysis of data owned by several sources. The protocol uses a semi-trusted third party and delivers on privacy and complexity.

1. Introduction

Regression is one of the most commonly used statistical tools in data analysis. It allows an investigator to model the relationship between a response variable and other variables that are assumed to influence such response. Response variables are usually indicators of performance and the other variables represent the attributes that affect this performance. For example, surgery completion time is an indicator of critical resources efficiency in hospitals [1]. Several studies looked for the attributes that cause these variation in surgery completion times [2]–[4], among the culprits, they found individual, team and organizational experience, learning curve heterogeneity and workload.

Regression modeling usually assumes that the data is freely accessible. However, in practice, data could be distributed among multiple different locations, and owned by different data holders.

Enormous utility could be gained by combining data from multiple sources; a regression performed on the union of the datasets may have better properties than a regression done on an individual dataset. For example, a regression analysis on integrated databases from multiple hospitals for surgery completion time is more powerful than regression analysis on an individual dataset.
However, the parties involved may not be willing to reveal their data as they might be competitors, or they may not mutually trust each other. Moreover, even if the data owners are willing to cooperate, legal and constitutional limitations would limit them from sharing some form of data such as personal health information [5].
A number of linear regression protocols that address these privacy concerns have been proposed in the literature; however, these protocols are either not completely private [6], [7] or are very demanding of the data holders involved in terms of complexity (extensive message passing among the participants and exponential computation complexity at each site) [8], [9].
We develop a privacy preserving linear regression protocol using a semi-trusted third party (the Evaluator). We show that our approach has several desirable properties. The complexity at each site is independent of the number of sites involved, and the complexity for the Evaluator is linear in the number of sites. Privacy of the health information is preserved and the statistical outcome retains the same precision as that of the raw data. Moreover, contrary to previous approaches, ours is complete [6]–[10]. It not only calculates the linear regression parameters for a fixed model, but also calculates the diagnostics for each model and presents a secure algorithm for selecting the model that has the best fit. These latter steps are in fact more important and more challenging than the calculation of the model parameters [5]. Two different model selection methods with varying levels of data sharing are presented: (i) The first method shares the integrated diagnostics for the models investigated, and (ii) the second method does not share the actual diagnostics, but only the maximal or minimal values (depending on the chosen diagnostic). An initial version of this work appeared in [11].

2. Linear Regression

2.1 Basic Concepts

Linear regression consists of modeling the relationship between a set of variables referred to as attributes (or independent variables) and a response variable (or output variable). It assumes that the relationship between the response variable and the independent variables is linear. Fitting a linear regression model consists of

a sequence of steps including estimation, diagnostics and model selection [12]. In what follows we give an overview of the steps involved, for more information the reader is referred to [12], [13]:
Assume that a dataset is composed of n vectors of input variables $\{x_1,...,x_n\}$ where $x_i = \{x_i^1,...,x_i^{|D|}\} \in \Re^{|D|}$ ($\Re$ is the set of real numbers and D is the set of attributes), and n output variables $y_1,...,y_n \in \Re$. We denote by X the $n \times D$ input matrix $[x_i] = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, and by Y the n output vector $Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$. Linear regression is the problem of finding the subset $d \subseteq D$ of the attributes that affect and shape the response variable Y, and then learning the function $f : \Re^{|d|} \to \Re$ that describe this dependency (of the output variables on the independent variables). Linear regression is based on the assumption that f is approximated by a linear map, i.e. $y_i \approx f(x_i) = \beta x_i^{|d|} + \beta_0 \;\; i \in \{1,...,n\}$ for some $\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \in \Re^{|d|}, \beta_0 \in \Re$, where $x_i^{|d|}$ is the vector x_i restricted to the set of attributes d in question. In linear regression literature, it is common to set $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix}$, and to augment every row $x_i^{|d|}$ with 1 (so $x_i^{|d|}$ is set to $[1 \;\; x_i^{|d|}]$). With that, the formulae above can be restated as: $y_i \approx f(x_i) = \beta x_i^{|d|}$, $i \in \{1,...,n\}$. In what follows, we abuse the notation and use the superscript d instead of $|d|$.
Given a subset of attributes $d \subseteq D$, to learn the regression model (i.e. to learn the function f) we need to find β such that $y_i \approx \beta x_i^{d}$ best fits the dataset. The difference between the actual value y_i and the estimated $\hat{y}_i = \beta x_i^{d}$ is referred to as the residuals: $\varepsilon_i = y_i - \hat{y}_i$. The goal in linear regression is to find β that

minimizes the square sum of the residuals ($\sum_{i=1}^{n} \varepsilon_i^2$). The method commonly used is referred to as the "least squares method" and is equivalent to solving the following equation:

$$\beta = (X^{d^T} X^d)^{-1} X^{d^T} Y \qquad (1)$$

2.2 *Model Selection*

Model selection (or variable selection) is the process of constructing a model that includes all relevant predicting variables. It is the process of determining the subset $d \subseteq D$ that best predicts the outcome variable Y. For model selection, we require (i) a goodness of fit measures (or model diagnostics) to assess how well a given model fits the data and (ii) an efficient mechanism to search within the space of available solutions (note that the size of the space is size $2^{|D|}$)

i. The most common goodness of fit measures are the Akaike Information criteria (or AIC), the Bayesian information criteria (or BIC) and the adjusted R^2 measure (or R_a^2). All these measures are expressed as a function of the sum of squares statistics, specifically the total sum of squares (SST) and the residual sum of squares (SSE) (refer to Table 1). The goodness of fit measures are defined as follows:

$$R_a^2 = 1 - \frac{(n-1)SSE}{(n-d-1)SST} \qquad (2)$$

$$AIC = n \log \frac{SSE}{n} + 2(d+1) \qquad (3)$$

$$BIC = n \log \frac{SSE}{n} + (d+1) \log n \qquad (4)$$

A goodness of fit measure in itself has no meaning. It makes sense when it is compared to the measures for the other models. The model with lowest AIC, lowest BIC or highest R_a^2 is a model with the least information loss and thus is considered the best approximation to the real model. However, the evaluation of a model (defined by a set d of predictors) is sometimes more involved. Analysts usually calculate and plot several different model diagnostics. Moreover, in the instance where two models have close diagnostics values (no single model is clearly the best), subsequent analysis is usually needed to determine the best model. For more information on these cases the reader is referred to [12], [13].

ii. Several strategies for iterating within the subset of predictors exist in the literature. These strategies could be either approximations or exact strategies. Approximation methods are computationally efficient but do not result in the selection of an optimal solution (or the "best" solution). Stepwise regression (that includes forward selection, backward elimination and bidirectional elimination) is a set of approximation methods. In forward selection the modeler starts with no variables in the model. Variables that improve the model the most (based on a pre-defined measurement criteria) are added one by one until no variable improves the model. Backward elimination works in the reverse way, and bidirectional elimination combines both approaches. Other approximation algorithms have a systematic way of travelling within a subset of the set $2^{|D|}$ of all possible models, the model with the best diagnostic (among the tested ones) is selected. The genetic algorithm [14] is one such example. Each candidate in the solution space is assumed to be a mix of desirable and non-desirable properties. The algorithm evolves from one solution to a better one by altering (mutating) some of the non-desirable properties.
Exact strategies evaluate all the $2^{|D|}$ possible models and result in the selection of the "best" solution, however, these are not computationally efficient [13].

Table 1. Summary Statistics, where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, and $\hat{y}_i = \beta x_i^d$

SSE	$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
SST	$\sum_{i=1}^{n}(y_i - \overline{y})^2$

3. Setting

In this paper, we consider the case of a dataset that is horizontally distributed among $k \geq 2$ data warehouses (or data owners). The different owners are interested in cooperatively studying the relationship between the independent and response variables, however they are not willing to share their data. The

relationship is assumed to be linear. This is sometimes referred to as privacy-preserving linear regression protocol [5]:
Let V be the matrix X augmented with column Y, i.e. $V=[X:Y]$. We consider the setting of k data holders, $DW_1 \ldots DW_k$, each holding part of the matrix V. The division is assumed to be horizontal, i.e. each party holds a subset of the records of V. Denote by V_i the subset of matrix V held by party i, then $V=\begin{bmatrix} V_1 \\ : \\ V_k \end{bmatrix}=\begin{bmatrix} X_1:Y_1 \\ : \\ X_k:Y_k \end{bmatrix}$. In what follows, we assume that $X_1=\begin{bmatrix} x_1 \\ : \\ x_{n_1} \end{bmatrix},\ldots,X_i=\begin{bmatrix} x_{n_{i-1}+1} \\ : \\ x_{n_i} \end{bmatrix}$ and $X_k=\begin{bmatrix} x_{n_{k-1}+1} \\ : \\ x_{n_k} \end{bmatrix}$. We denote by $SSE_i=\sum_{j=n_{i-1}+1}^{n_i}(y_j-\hat{y}_j)^2$ and $SST_i=\sum_{j=n_{i-1}+1}^{n_i}(y_j-\bar{y})^2$ the local statistics for site i, i.e. $SSE=\sum_{i=1}^{k}SSE_i$ and $SST=\sum_{i=1}^{k}SST_i$

Before proceeding with an overview of the algorithm, we present two important properties for the horizontally distributed data:

1. For any $d \subseteq D$, $X^{d^T}X^d$ can be extracted from X^TX by removing all entries ij from matrix X^TX where either i or j do not represent a variable in d. The same applies for $X^{d^T}Y$.
2. Given that $n>|D|$, for any $d \subseteq D$ we have that:
   $X^{d^T}X^d=\sum_{i=1}^{k}(X^d{}_i{}^T X^d{}_i)$ and that $X^{d^T}Y=\sum_{i=1}^{k}X_i{}^{d^T}Y_i$.
   Hence Equation (1) is equivalent to:
   $$\beta=(\sum_{i=1}^{k}(X_i{}^{d^T}X_i{}^d))^{-1}(\sum_{i=1}^{k}X_i{}^{d^T}Y_i) \qquad (5)$$

In what follows, we present a privacy-preserving linear regression protocol that uses a third party. The third party is referred to as the Evaluator. The Evaluator is assumed to follow the protocol correctly however if some data holders are corrupt, then the Evaluator will collaborate with them to obtain sensitive information about the data. Similarly, it is assumed that a corrupt data holder will correctly follow the protocol. We assume that up to $l-1$ data holders can be corrupt for some

$0 < l < k$, thus, if $l = 1$ then all data holders are honest. The protocol is composed of 3 functions:
(a) Pre-computations, (b) a core regression protocol (referred to as CP), and (c) an iterative protocol referred to as IP:
The pre-computations are done once at the beginning of the algorithm, then the IP protocol runs. It's role is to iterate over different values of $d \subseteq D$, calling CP for each such value of d. The CP protocol is performed by the evaluator with the collaboration of l out of the k data warehouses. CP takes as input a subset $d \subseteq D$ and computes β and a model diagnostic for that given d.
We consider two different scenarios related to model diagnostic:

1. Every time a model diagnostic is securely calculated for a set of predictors d, its value is shared among the different parties, or
2. The diagnostics are securely calculated for the different set of predictors d but they are not shared. Whenever needed, the maximum (or minimum) among the encrypted values of a set of diagnostics can be securely calculated. That information is used to decide the best model in the set.

The scenarios above have implications on the IP protocol that can be used. If model diagnostics are disclosed to the different parties at every iteration, then their values may be used in guiding future iteration steps. Otherwise, if model diagnostics are not shared, then the model selection should be solely based on the order of the diagnostics [13].
In this paper, we will assume an all possible-subsets *IP* protocol, i.e., for every possible model (among the $2^{|D|}$ models) we find its corresponding β, and either $R_a^{\ 2}$, AIC, or BIC (note that this can easily be substituted with another more efficient *IP* protocol). If model diagnostics are not to be shared, then the secure maximum (or minimum) is executed at the end after all diagnostics are securely calculated.

An outline of the algorithm is presented in Figure 2, where the IP function and the algorithm flow are presented in details. The remaining functions will be presented in details in Section 6. For more information on regression the reader is referred to [5], [13].

4. Related Work

A number of protocols have been proposed for the collaborative computation of linear regression when data is horizontally distributed among different parties [6]–[10], [15]–[19]. These protocols respond to different levels of data privacy requirements (refer to Figure 1). The approximate approaches, displayed on the left

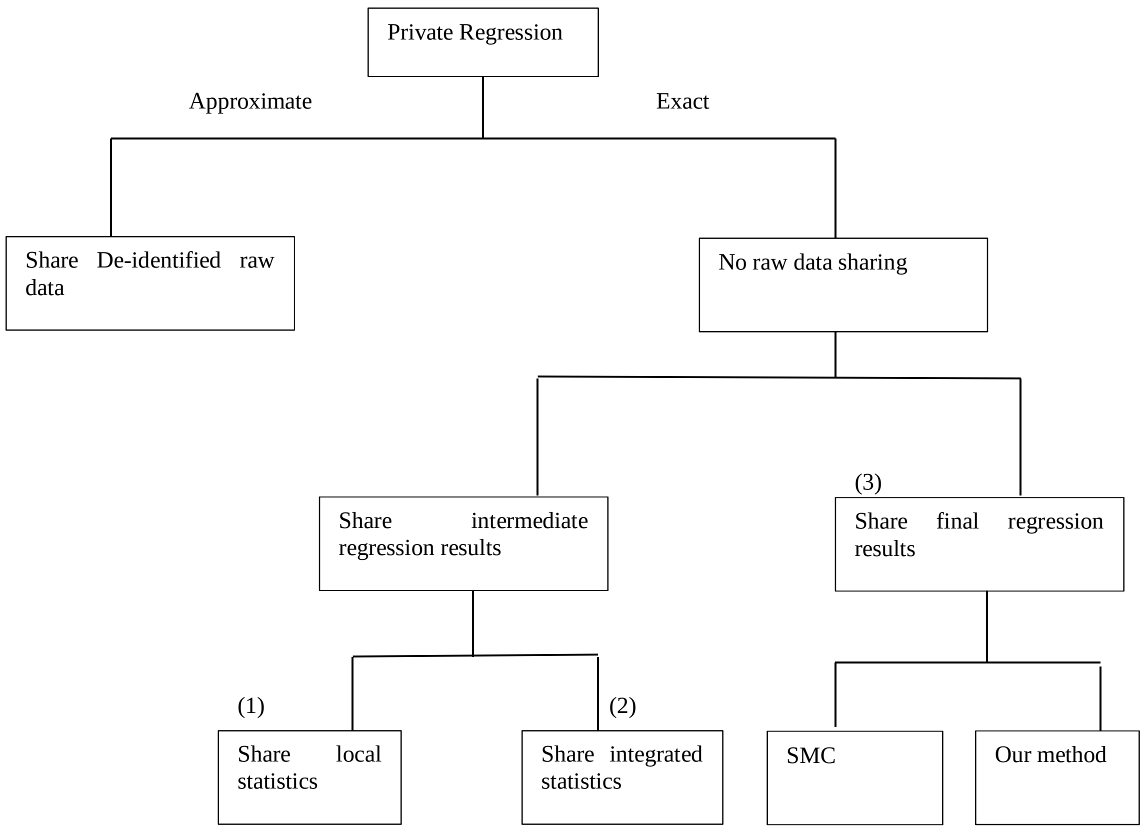

Figure 1. A hierarchical decomposition of the different private regression algorithms based on the type/amount of shared data

side of Figure 1, are based on de-identification. The different parties collectively de-identify their local dataset using anonymization techniques and secure multiparty computation protocols [16]–[18], [20]. An integrated dataset is built from the individual de-identified datasets and regression protocols are applied to the shared de-identified data. Although this approach is computationally efficient, its utility is questionable and is a function of the de-identification applied to the data. The exact approaches displayed on the right side of Figure 1 perform regression on exact data. We classified these approaches according to the type/amount of data shared to perform regression. In some approaches, intermediate regression results are shared among the different parties (1) These could either be local statistics (i.e. ${X_i}^T X_i$, ${X_i}^T Y_i$, SSE_i and SST_i) or (2) integrated statistics (i.e. $X^T X, X^T Y, SSE$, and SST). While in other approaches (3) only regression parameters are shared among the different parties (i.e. β and model diagnostics):

1. Sharing local statistics: This approach was first introduced by Du et al in [7]. They suggest sharing local aggregate information. Each site i shares with the other sites the aggregate values: ${X_i}^T X_i$ and ${X_i}^T Y_i$. This way, each site first adds the aggregate information to obtain $X^T X$ and $X^T Y$, finds the inverse of $X^T X$, then uses Equation (1) to estimate the parameters β of the regression. To calculate model diagnostics, the sites share their respective local summary statistics SSE_i and SST_i, and uses these to calculate SSE and SST. This method, although efficient, was criticized for being non-private as identifying information can be embedded in the shared local statistics [5], [8].
2. Sharing integrated statistics: Another protocol due to Karr et al [6] suggests using secure multiparty computation to securely compute the sum of the local statistics: ${X_i}^T X_i$ and ${X_i}^T Y_i$. The final sums $X^T X$ and $X^T Y$ are then shared among the different sites. The same applies for local summary statistics, i.e. a secure sum is performed for SSE_i and SST_i, the final sums, SSE and SST, are shared among the different sites. This protocol, although efficient, was also deemed to be non-private. The reader is referred to [8] for examples of inappropriate disclosure from matrices $X^T X$ and $X^T Y$.
3. Sharing final regression results: We found three protocols under this category. Two of these protocols [8], [9] use secret sharing and homomorphic encryption to privately calculate $X^T X$, $(X^T X)^{-1}$ and $X^T Y$, and then to securely multiply $(X^T X)^{-1}$ and $X^T Y$. Both solutions make heavy usage of secure multiparty computation. As such, all data holders must remain online throughout the entire procedure. In both of these

protocols, the main computational component is the secure inversion of $(X^T X)$ and its extensive usage of the secure multiparty matrix multiplication protocol of [21] extended to k parties. Each use of this k-party multiplication requires each pair of participants to execute a 2-party secure matrix multiplication protocol. This amounts to a total of $\frac{k(k-1)}{2}$ multiparty matrix multiplications. Such a k-party multiplication has each party executing a combination of k homomorphic matrix multiplications and encryptions-decryptions under Paillier cryptosystem, as well as sending k matrices to the other parties.

The third protocol was presented in [10], it uses additive encryption and Yao Garbled circuits. The protocol uses two non-communicating semi-trusted third parties. One party executes the algorithm, while the other holds encryption keys and generates garbled inputs. The additive encryption is used to privately compute $X^T X$ and $X^T Y$, and Yao Garbled circuits to privately find the inverse of $X^T X$. While this solution does not require the involvement of all data holders, it requires two non-colluding semi-trusted parties each sharing part of the output. Moreover, the protocol requires the construction of garbled circuits. The construction of such circuits as well as its theoretical complexity were not tackled in the paper.

The protocols discussed under this category are **incomplete**. They do not tackle the more important and challenging steps of the secure calculation of model diagnostics and of the selection of the final model. In fact they only present a way to calculate and share β.

The algorithm presented in this paper falls under the third category, thus only the final regression results, β, are shared among the different parties. However, in addition to calculating the linear regression parameters β of a fixed model, the protocol calculates model diagnostics. As discussed earlier, for the calculation of model diagnostics we present two options: (i) the different parties either share the integrated model diagnostics, i.e. R_a^2, BIC, or AIC, or (ii) the parties can decide to go one step further in limiting data sharing and keep the integrated diagnostics for the different models private and disclose only the maximal (or minimal) value among all calculated diagnostics. A secure maximum/minimum protocol is presented for that purpose. Model selection in that case is based on the model with maximal AIC, maximal BIC or minimal R_a^2.

5. Public Key Cryptosystem

In Section 6 we present the privacy preserving regression protocol, but first, in this section, we introduce the properties of the public key cryptosystems that are used in our protocol.

Given a message m_i, we denote its ciphertext by $c_i = Enc_{pk}(m_i)$ where pk is the public key used in the encryption. A ciphertext c_i can be decrypted using a secret key sk and a decryption function, $m_i = Dec_{sk}(c_i)$. We will simply use $Enc(m_i)$/ $Dec(c_i)$ when pk / sk is clear from the context. In our protocol, we use Paillier cryptosystem [22] for the case where $l = 1$ and threshold Paillier cryptosystem [23] when $l > 1$ for their useful homomorphic properties.

Paillier cryptosystem is additively homomorphic, as such the sum of two messages can be obtained from their respective cyphertexts. For Paillier, this translates to $Enc_{pk}(m_i + m_j) = Enc_{pk}(m_i) \times Enc_{pk}(m_j)$ [22]. Moreover, Paillier allows a limited form of homomorphic multiplication, in that we can multiply an encrypted message by a plaintext. It is done as follows: $Enc(m_i)^{m_j} = Enc(m_i m_j)$. With these properties, one can subtract two messages via their cyphertexts as follows: $Enc(m_i - m_j) = c_i {c_j}^{-1}$, and divide any message homomorphicaly as follows: $Enc\left(\frac{km_i}{k}\right) = Enc(km_i)^{-k}$.

To simplify notation, given a matrix M, we let $Enc(M)$ denote the entry-wise encryption of M. Thus, given two matrices A and B, the two properties of the Paillier encryption allows us to calculate the encrypted product $Enc(AB)$ from $Enc(A)$ and B as follows: $Enc(AB)_{ij} = \prod_k Enc(A)_{ik}{}^{B_{kj}}$, where M_{ij} represents the ij^{th} entry in Matrix M. Similarly, $Enc(AB)$ can be calculated from A and $Enc(B)$.

In a (n,t)-threshold cryptosystem, the secret decryption key is distributed among n different entities such that a subset of at least t of them are needed to perform the decryption [23], [24]. I.e. in order for the decryption to occur, at least t parties have to correctly perform their share of the decryption. The decryption shares are then combined to obtain the final decryption.

Note that our protocol will be using the threshold Paillier [23] cryptosystem when $l > 1$. This can be set up through a trusted party that will generate and distribute the public and secret keys. The trusted party can then erase all information pertaining to the key generation. If no such trusted party is available, the keys can be generated using secure multiparty computations [25]. Although this requires more computation overhead from each data owner, it only has to be done once. As such, it is an acceptable tradeoff.

In an (n,t)-threshold Paillier cryptosystem, the encryption is identical to the regular scheme. For the decryption, each party involved is required to compute the exponentiation of the ciphertext by their secret key. The product of these shares is then computed to proceed with the decryption. Since the validity of the decryption depends on the validity of the shares, threshold decryption protocols involve proofs of knowledge between each participant to prevent attacks by malicious parties. The complexity of the decryption is thus dominated by these proofs of knowledge [23].

We note that in our setting, each data owner will correctly execute the protocol even if they are corrupt, since they genuinely want the correct result. As such, we do not require the proofs of knowledge. This makes the threshold decryption only slightly more complex than the decryption in the setting $l = 1$.

6. Protocol

In what follows, we present the Pre-computation function (also referred to as Phase 0), as well as the CP function which is composed of two phases, Phase 1, and Phase 2:

- In Phase 0 some pre-computations are done. These are the computations that do not depend on β, thus they can be computed once at the beginning of the Protocol. They are $X^T X, X^T Y,$ and ST.
- The CP protocol is executed several times for different subsets $d \subseteq D$, it computes β and a diagnostic for the given d with the collaboration of l out of k data warehouses, say $DW_1,\ldots,DW_l$. Phase 1 of CP is dedicated to the calculation of the regression coefficients β and Phase 2 is dedicated for the calculation of model diagnostics.

We assume that all the inputs are integer valued, due to the use of Paillier's cryptosystem. This is not a problem, as the data owners can multiply their data by a large non-private number. The effects of this multiplication can then be removed in intermediate/final results [21], [25].

```
Function PreComputation()
{
        Phase 0
}

Function CP(d)
{
        Phase 1        # Calculates β for the model
                        with attributes d .
        Phase 2        # Calculates Ra^2 for the model
                        with attributes d .
}

Function IP(d)
{
 V_{Ra^2} ← φ        #Diagnostics vector
 V_β ← φ             #Parameters' vector
For every d ∈ P^D           # Power set of D

        {
         CP(d)          # Calculates β and Ra^2 for the
                          model with attributes d.
         Append Ra^2 to V_{Ra^2}
         Append β to V_β
        }
SecureMin(V_{Ra^2} )
}
```

Figure 2. Algorithm flowchart assuming $R_a^{\ 2}$ diagnostic

Before presenting the protocol, we first start with some basic functions used throughout the protocol. The protocol will be presented for the general case where $l > 1$. When $l = 1$, some steps can be optimized to slightly reduce the number of messages sent. These steps are presented after the protocol in a separate section. We assume that the total number of records n is public knowledge.

6.1 Functions

6.1.1 Basic functions

The protocol uses several basic functions:

Creating Random Matrices of size $d\times d$, or *CRM*(d): $DW_1,...,DW_l$ and the Evaluator each generate a secret random $d \times d$ matrix $B_1,...,B_l, A$ respectively. We denote by B the product $B_1B_2...B_l$.

Creating Random Integers, or *CRI*: $DW_1,...,DW_l$ each generate a secret random integer $b_1,...,b_l$ respectively, while the Evaluator generates two random secret integers a and r. We denote by b the product $b_1b_2..b_l$.

Encryption Function for matrices, or $Enc(M)$, encrypts the entries of a matrix M and send the results to the Evaluator. This is an extension of the regular encryption function on integers.

Decryption Function for matrices, or $Dec(Enc(M))$, decrypts the entries of the matrix $Enc(M)$ and send the results to the Evaluator. This function requires the involvement of l data warehouses as well as the evaluator. The evaluator sends the matrix $Enc(M)$ to $DW_1,...,DW_l$, each of the data warehouses does its own share of decryption and sends the result to the Evaluator. The Evaluator combines the results to obtain $M = Dec(Enc(M))$.

Right Matrix Multiplication Sequence Function, or *RMMS*($Enc(M)$), computes $Enc(MB)$. The Evaluator sends $Enc(M)$ to DW_1, who uses it to homomorphically compute $Enc(MB_1)$ using it secret matrix B_1. The result is then sent to DW_2, who in turn computes $Enc(MB_1B_2)$. The process repeats with $DW_3,...,DW_l$, and the result $Enc(MB)$ is sent back to the evaluator.

Left Matrix Multiplication Sequence Function, or *LMMS*($Enc(M)$), computes $Enc(BM)$ It is similar to *RMMS*($B, Enc(M)$), but the order on the data warehouse is reversed.

Integer Multiplication Sequence Function, or *IMS*($Enc(m)$), the Evaluator sends an Encrypted value $Enc(m)$ to DW_1, who uses it to calculate the encrypted product $Enc(mb_1) = Enc(m)^{b_1}$, using it secret integer b_1. The result is then sent to DW_2, who in turn computes $Enc(mb_1b_2)$. The process repeats with $DW_3,\ldots,DW_l$, and the result $Enc(mb)$ is sent back to the evaluator.

Integer Removal Sequence Function, or *IRS(* $Enc(bm)$ *)*, the Evaluator sends an Encrypted value $Enc(bm)$ to DW_1, who uses it to calculate the encrypted product $Enc(bm/b_1) = Enc(bm)^{1/b_1}$, using its secret integer b_1. The result is then sent to DW_2, who in turn computes $Enc(bm/(b_1b_2))$. The process repeats with $DW_3,\ldots,$ DW_l, and the result $Enc(m)$ is sent back to the evaluator.

6.1.2 Secure maximum/minimum protocol

In order to calculate the maximum/minimum diagnostic value privately, it is enough to come up with a protocol that privately compares two encrypted values. In what follows, we present a protocol that privately compares two encrypted values m_1 and m_2, the protocol is executed by the evaluator with assistance from $DW_1,\ldots,DW_l$:

1. The evaluator computes $Enc(m_1 - m_2) = Enc(m_1)Enc(m_2)^{-1}$ and calls for *CRI*. Thus the evaluator and every site produce a random secret integer.
2. The evaluator calls for *IMS(* $Enc(a(m_1 - m_2))$ *)*, and obtains $Enc(ba(m_1 - m_2))$.
3. The Evaluator initiates $Dec(Enc(ba(m_1 - m_2)))$, if the returned result $ba(m_1 - m_2)$ is negative, then $m_1 < m_2$ otherwise, $m_1 \geq m_2$.

Special considerations are to be taken depending on the encryption scheme used. In the case of a Paillier encryption scheme with public key *N*, the result will only be valid if $|ba(m_1 - m_2)| < N/2$. This can be accomplished if $|m_1|$ and $|m_2|$ are bounded above by some value $U/2$ and the obfuscating values are chosen such that $ba < N/2U$.

6.2 Phase 0: Precomputations

At the beginning of this Phase, the Evaluator initiates *CRI*, thus the l data warehouses as well as the Evaluator generate a secret random integer each. This phase is composed of two main computations:

- **Computations of** $Enc(X^T X)$ **and** $Enc(X^T Y)$: Each data holder DW_i locally computes her full local matrices $X_i^T X_i$, and $X_i^T Y_i$. She encrypts the matrices and sends them to the Evaluator. The Evaluator performs homomorphic additions and obtains $Enc(X^T X) = Enc(\sum_{i=1}^{k} X_i^T X_i)$, and $Enc(X^T Y) = Enc(\sum_{i=1}^{k} X_i^T Y_i)$.
- **Computation of Enc(*SST*)=*Enc*(** $\sum_{j=1}^{n} (y_j - \bar{y})^2$ **)**: Note that $\sum_{j=1}^{n} (y_j - \bar{y})^2 = \sum_{j=1}^{n} {y_j}^2 + \sum_{j=1}^{n} \bar{y}^2 - \sum_{j=1}^{n} 2y_j\bar{y}$. Let $\delta_i = \sum_{i=n_{j-1}+1}^{n_j} 2y_j\bar{y}$ and $\alpha_i = \sum_{i=n_{j-1}+1}^{n_j} {y_j}^2$. The computation proceeds as follows:
  - Each data warehouse DW_i sends their encrypted local aggregate $\frac{\phi_i}{n} = \sum_{j=n_{i-1}+1}^{n_i} \frac{y_j}{n}$ to the Evaluator.
  - The Evaluator homomorphically adds these values to get $Enc(\bar{y}) = Enc(\sum_{j=1}^{n} \frac{y_j}{n})$, and then calculates $Enc(a\bar{y}) = Enc(\bar{y})^a$.
  - The Evaluator initiates $IMS(Enc(a\bar{y}))$ and receives $Enc(ba\bar{y})$. It then initiates $Dec(Enc(ba\bar{y}))$ and gets $ba\bar{y}$.
  - The Evaluator computes $Enc(b^2 a^2 \bar{y}^2)$ and initiates 2 consecutive calls to IMS($Enc(b^2 a^2 \bar{y}^2)$) which results in $Enc(a^2 \bar{y}^2)$. The Evaluator then computes $Enc(\bar{y}^2)$.
  - The Evaluator propagates $Enc(a\bar{y})$ to all data warehouses.
  - Each data warehouse computes $Enc(a\delta_i) = Enc(\sum_{j=n_{i-1}+1}^{n_i} (2ay_j\bar{y})) = \prod_{j=n_{i-1}+1}^{n_i} Enc(a\bar{y})^{2\sum y_j}$ and $Enc(\alpha_i) = Enc(\sum_{j=n_{i-1}+1}^{n_i} {y_j}^2)$. Both are sent back to the Evaluator.

- The Evaluator computes $Enc(\sum_{i=1}^{k} a\delta_i)$ and recovers $Enc(\sum_{i=1}^{k} \delta_i)$. The Evaluator then proceeds to compute $Enc(SST) = Enc(\sum_{i=1}^{k} \alpha_i - \sum_{i=1}^{k} \delta_i + n\bar{y}^2) = Enc(\sum_{j=1}^{n} (y_j - \bar{y})^2)$.

6.3 *Protocol:* $CP(d)$

First, given d, the evaluator extracts the encryptions of $Z' = X^{d^T} Y$ and $Z = X^{d^T} X^d$ from $Enc(X^T X)$ and $Enc(X^T Y)$ respectively. Then the Evaluator initiates *CRM*(d) and *CRI*. These are necessary at every iteration of the protocol because the random integers and random matrices need to be refreshed for every value d.

6.3.1 Phase 1: Computing β

In this phase of the protocol, the Evaluator needs to compute $Z^{-1}Z'$. The steps are the following:

- The Evaluator computes $Enc(ZA)$, initiates *RMMS*($Enc(ZA)$) and receives $Enc(ZAB)$.
- The evaluator initiates $Dec(Enc(ZAB))$ and receives ZAB.
- The evaluator computes $B^{-1}A^{-1}Z^{-1} = (ZAB)^{-1}$ and calculates $Enc(B^{-1}A^{-1}\beta)$ using $B^{-1}A^{-1}Z^{-1}$ and $Enc(Z')$. (note that if the matrix $B^{-1}A^{-1}Z^{-1}$ has non-integer values, then it can be multiplied by a large integer value, The effect of this integer can be removed from the final outcome β).
- The Evaluator obtains $Enc(A^{-1}\beta)$ by initiating *LMMS*($Enc(B^{-1}A^{-1}\beta)$) and computes $Enc(\beta)$ homomorphically.
- The Evaluator initiates $Dec(Enc(\beta))$, recovers β and sends it to all data warehouses.

6.3.2 Phase 2: Model Diagnostics

In this section we show how $Enc(SSE)$ can be securely calculated. Then we illustrate how $Enc(R_a^2)$ can be calculated using $Enc(SSE)$ and $Enc(SST)$. $Enc(AIC)$ and $Enc(BIC)$ could be calculated using similar procedures.

1. Computing *Enc(SSE)* for a Given *d*

As each data warehouse DW_i recieves β, they calculate their local residuals: $\zeta_i = \sum_{j=n_{i-1}+1}^{n_i} (y_j - \hat{y}_j)^2$, encrypt it and send it to the Evaluator. The Evaluator then adds the local residuals homomorphically to obtain $Enc(SSE) = Enc(\zeta_1) \times ... \times Enc(\zeta_k)$.

2. Computing model diagnostics for a Given *d*

In this subsection we illustrate how $Enc(R_a^2)$ could be securely calculated using $Enc(SSE)$ and $Enc(SST)$. Other model diagnostics can be calculated using similar procedures:

- The Evaluator computes $Enc(rSSE)$ and $Enc(aSST)$. He then initiates $IMS(Enc(rSSE))$ and $IMS(Enc(aSST))$ and receives $Enc(brSSE)$ and $Enc(baSSE)$ respectively.
- The Evaluator then initiates a decryption round for $Enc(brSSE)$ to obtain $brSSE$, and uses it to compute $baSSE = brSSE \frac{a}{r}$.
- The Evaluator now calculates $Enc(R_a^2)$ homomorphically. Assuming that C is a very large public integer, the computation proceeds as follows:
  - o Evaluator computes $\frac{n-1}{(n-d-1)baSSE}$ converts it into an integer $C\frac{n-1}{(n-d-1)baSSE}$ then
  - o Evaluator calculates $Enc(CR_a^2) = Enc(C)Enc(ba\alpha)^{-C\frac{(n-1)}{(n-d-1)ba\zeta}}$
- (Optional Step) The Evaluator initiates $Dec(Enc(\frac{SSE}{SST}))$ and propagates the result to the different warehouses so that $R_a^{\ 2}$ can be calculated with the public values n and d.

6.4 Choosing the "best" model

In order to choose the model that best fits the data, we present two options:

1. Every time a model diagnostics is calculated, it can be decrypted by calling the *Dec* function, and then shared among all sites (last optional step of Section 6.5.2). or

2. We can wait until the encrypted diagnostics of all the different models are calculated, and we use our private minimum/maximum algorithm to find the model with the minimal AIC/BIC or the maximal R_a^2 (depending on which diagnosis we desire). The found model is designated as the best fit.

6.5 *Special Considerations for the case l=1*

For the case $l = 1$, all the data owners are assumed to be incorruptible and all the decryption and obfuscation is delegated to one data warehouse, say DW_1. As such, the steps that initiate a multiplication sequence (*RMMS*, *LMMS* or *IMS*) followed by a decryption can be reversed and merged. In other words, DW_1 can do the decryption first followed by the multiplication by its random number.

For example, in the computation of $Enc(SST)$ in Phase 0, steps 3 and 4 can be replaced by "The Evaluator sends $Enc(a\overline{y})$ to DW_1, who decrypts to obtain $a\overline{y}$. DW_1 then compute and send $ba\overline{y}$ back to the evaluator." This will considerably reduce the complexity of DW_1's computations when working with matrices.

Note that the role of DW_1 can be assumed by another semi-trusted third party (STTP) if available. In such case, the Evaluator and the STTP will together compute the CP protocol. Both third parties should be trusted to follow the protocol correctly and to not communicate secretly outside the protocol.

6.6 *Protocol Modification*

In order to execute our linear regression protocol, the l participating data warehouses have to be online throughout the whole process. It would be ideal if the remaining $k-l$ data warehouses could send their data at Phase 0, then stay offline throughout the remaining protocol. However this is not that case, these data warehouses have to participate in the calculation of SSE at each iteration of the CP protocol. With some changes to the protocol, it is possible for the data warehouses to send their fully encrypted matrices $Enc(X_i)$ and $Enc(Y_i)$ to the Evaluator at the start of the protocol (i.e. in Phase 0) then stay offline for the whole process afterwards. The Evaluator can use these encrypted matrices to calculate $Enc(SSE)$ without the involvement of the $k-l$ data warehouses. In other words, for a given d subset of D, and for every data warehouse i, the Evaluator can calculate $Enc(SSE_i)$ using $\beta, Enc(X_i)$, and $Enc(Y_i)$ as follows:

For every $j \in \{n_{i-1}+1, \ldots, n_i\}$, we calculate:

$$Enc(\hat{y}_j) = \prod_{l \in d} Enc(x_{jl})^{\beta_k} = Enc(\beta x_j^{\,d}),$$

then, given $Enc(\hat{y}_j)$ and $Enc(y_j)$, we can use a series of *CRI*, *Dec* and *Enc* functions to calculate $Enc(\hat{y}_j^{\,2})$, $Enc(y_j^{\,2})$ and $Enc(\hat{y}_j y_j)$, which in turn can be used to calculate $Enc(SSE_i) = \sum_{n_{i-1}+1}^{n_i} Enc(\hat{y}_j^{\,2}) + Enc(y_j^{\,2}) - Enc(\hat{y}_j y_j)^2$.

The problem with this modification is that the data warehouses would give out their local number of records, $n_1,..,n_k$. If this is considered private information, then the original protocol has to be followed. Note that this modification requires considerably more space and complexity at the Evaluator side as the k encrypted matrices have to be stored and then used to calculate $Enc(SSE_i)$.

7. Privacy Discussion

In this section we study the privacy of our protocol and show that no party can learn any information other than the final results of the regression. We assume that $l-1$ parties are corruptible and that the total number of records, n, is public knowledge.

Since Paillier is semantically secure [22], then no information can be gained from the ciphertexts exchanged throughout the protocol unless they are decrypted. However, since l parties are required for a successful decryption, then the $l-1$ corrupted parties cannot perform decryption on their own, as such, we will only investigate privacy threats due to the decrypted values each party obtains.

In Phase 0, all the values are encrypted except for $ba\overline{y}$. All l active data owners and the evaluator have access to this value. If the corrupted data owners and the evaluator collaborate, they can remove a and at most $l-1$ of the random numbers added. In other words, they can reduce the value to $b'\overline{y}$, where b' is the random unknown integer associated with the honest data warehouse. As b' is unknown, no information about $\overline{y}$ can be recovered (note that In the case where $l=1$, the active data owner obtains $a\overline{y}$, but since a is random and since DW_1 is incorruptible, no information about $\overline{y}$ can be gained).

The same reasoning applies for Phase 2, where all the information is encrypted except for $brSSE$, which is always obfuscated by at least one random integer (for

the case where $l=1$, DW_1 can also obtain $aSST$ but no information about SSE or SST can be gathered since both are obfuscated by different random integers).
In Phase 1, all the matrices are encrypted except for ZAB that all active parties and the evaluator compute. Assuming that DW_i is the incorruptible party, then the evaluator and the corrupted parties can calculate $ZAB'B_i$, where $B'=B_1B_2...B_{i-1}$, B' is known to them but not Z or B_i. Thus, since B_i is random, the evaluator cannot recover Z, even while knowing $AB_1B_2...B_{i-1}$.
Thus, since all values in the protocol are either encrypted or obfuscated by some random values, no party can learn additional information from the protocol other than the final results of the computations.

8. Complexity

In this section, we evaluate the computational complexity of one protocol iteration. We will evaluate the individual burden on each participating party as well as the total complexity of the protocol. We abuse the notation and denote by d the number of attributes in an iteration and by D be the total number of attributes in the database. A message will be considered either as one value, encrypted or not.
The complexity will be expressed in terms of some basic functional units. These units are homomorphic multiplication (HM) and homomorphic addition (HA), where (assuming that we are using an instance of Paillier with modulus m^2) HA is equivalent to multiplying two integers modulo m^2, and HM is equivalent to computing an exponentiation modulo m^2 where the exponent is at most m. As such, 1 HM is equivalent to $\log(m)$ HA [26].
With the above considerations, It follows that a message encryption is equivalent to 2HM and 1HA (thus it is dominated by 2HM), while a standard decryption is essentially 1HM [22]. As for a (k,l)-threshold decryption, it is equivalent to having each of the l involved parties compute one decryption and the evaluator to compute l HA's (for the product of the encrypted shares). As such, we can reasonably assume that (k,l)-threshold decryption is bounded above by a total computational complexity of lHM, making it only slightly more expensive than standard decryption.

We will start by evaluating the complexity of the basic functions used throughout the protocol.

- $Enc(M)$: If M is a $d \times g$ matrix, then the function involves dg encryptions
- $Dec(M)$: If M is a $d \times g$ matrix and since this function is performed by l data warehouses, it involves dgdecryptions and dg messages per data warehouses, l Has and dg messages for the evaluator.
- RMMS($Enc(M)$) and LMMS($Enc(M)$): These functions are performed collectively by the l data warehouses on square matrices. If M is a $d \times d$ matrix, then to execute this function, each DW has to send d^2message to another DW and perform, at most, d^3HM. That makes a total of ld^2 messages and at most ld^3 HM operations.
- IMS($Enc(m)$) and $IRS($ $Enc(bm)$): Each party sends one message to exactly one other party, and each participating data warehouse has to compute one HM. That makes a total of l messages and lHM.
- Secure Min/Max protocol: Step 1 of the protocol requires 1HM, 1HA and one message from the evaluator. Step 2 requires 1HM and one message from each of the l data warehouses. Step 3 requires 1HM per data warehouse, 1 message per party, and lHAs from the evaluator. Thus the overall protocol is governed by $2t$HM and $2t$ messages from each data warehouse, while the evaluator needs tHM and $2t$ messages, where t is the number of models checked.

The complexity for the different phases of the protocol are depicted in Tables 2, 3 and 4 below. The tables show the computational complexity as well as the number of messages performed by the Evaluator, by a regular DW and by a participating DW. Since HM is the dominant operation, the computational complexity is governed by the number of HM operations.

Table 2. Complexity of Phase 0

Phase 0	Evaluator	Regular DWs	l DWs
HM operations	O(1)	O(d^2)	O(d^2)
Messages	O(k)	O(d^2)	O(ld^2)

Table 3. Complexity of Phase 1

Phase 1	Evaluator	Regular DWs	l DWs
HM operations	O(d^3)	0	O(d^3)
Messages	O($kd + d^2$)	0	O(ld^2)

Table 4. Complexity of Phase 2

Phase 2	Evaluator	Regular DWs	l DWs
HM operations	O(1)	O(1)	O(1)
Messages	O(k)	O(1)	O(l)

Note that Phase 0 is performed once at the beginning of the protocol, while Phases 1 and 2 iterate over subsets d of D. Table 5 presents the complexity of the one iteration of the protocol.

Table 5. Complexity of one iteration

Phase 2	Evaluator	Regular DWs	l DWs
HM operations	O(d^3)	O(1)	O(d^3)
Messages	O($kd + d^2$)	O(1)	O((ld^2)

Note that, with the protocol modification suggested in Section 6.7, the evaluator would perform additional $O(nd)$ HM operations, and the participating DWs would perform additional $O(n)$ HM operations. However, the regular DWs would only participate in Phase 0 and remain idle in all iterations afterwards.

As can be seen from this evaluation, the total complexity of the protocol per iteration is linear in k. The complexity of the Evaluator and the participating data warehouses depends only on the size d of the matrices. This shows that our protocol allows the data owners to greatly reduce the computational power needed for a multiparty regression by making use of a STTP (the Evaluator).
For the sake of comparison, we take a closer look at the complexity of the schemes presented in [9] and [8] at the level of each individual participants. For this we shall look mostly at the secure multiparty matrix multiplication protocol of [21] (which was followed in [9] and [8]). In the 2-party case, one party has to compute about $3d^2$ HM operations while the second party has to execute about d^3 HM operations. In the k-party protocol an average of kd^3 HM operations is performed for each participating member.
This multiparty secure matrix protocol is executed at least 2 times in [8] and up to 248 times in [9] when computing the inverse of Z. We note that, for any k, our complete protocol $CP(d)$ involves less computational burden and messages for each party than a single matrix inversion in [8] or [9]. This is due to the fact that the individual complexity in our protocol is independent of k for all participants but the Evaluator.

9. Discussion

We presented a practical system that performs secure linear regression on horizontally distributed data. The different data holders do not share anything about their data apart from the regression parameters and the model diagnostics.
Different from existing secure approaches, our approach is complete. It not only calculates the parameters β of a fixed model, but also includes model diagnostics and selection, which are more important and more challenging steps [5]. Our model is superior in terms of complexity on the data holders end as the Evaluator absorbs most of the regression complexity.
However the problem of secure regression is not completely solved. We still do not have a way to securely check the linearity property. In fact, if linearity holds at each site separately then it may not hold for the union of the data. On the other hand we still need to generalize the protocol to cover generalized linear models (such as Poisson regression).
In addition to the above issues, extensions to this work would tackle issues related to data overlap between sites, missing data and measurement errors in a secure manner [9].
On the other hand, we are currently in the process of applying the protocol on the union of three datasets from the state of Pennsylvania (over 1.5 million records). The study aims to find the attributes that affect surgery completion times and come up with recommendations. The trusted third party (the Evaluator) is the IBM Cloud located at Western University.

Acknowledgments

The research is supported in part by Sidra Medical and Research Center and by IBM Canada Research and Development Center (ibm.com), the Southern Ontario Smart Computing Innovation Platform (www.soscip.org).
We would like to thank Dr. Mohammed El Anbari for his valuable comments on previous versions of the manuscript.

References

[[1] E. M. Stahl, Emergency Department Overcrowding: Its Evolution and Effect on Patient Populations in Massachusetts. ProQuest, 2008.

[2] D. S. Kc and C. Terwiesch, “Impact of workload on service time and patient safety: An econometric analysis of hospital operations,” Manag. Sci., vol. 55, no. 9, pp. 1486–1498, 2009.
[3] G. P. Pisano, R. M. Bohmer, and A. C. Edmondson, “Organizational differences in rates of learning: Evidence from the adoption of minimally invasive cardiac surgery,” Manag. Sci., vol. 47, no. 6, pp. 752–768, 2001.
[4] R. Reagans, L. Argote, and D. Brooks, “Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together,” Manag. Sci., vol. 51, no. 6, pp. 869–881, 2005.
[5] J. Vaidya, C. W. Clifton, and Y. M. Zhu, Privacy Preserving Data Mining. Springer, 2005.
[6] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, “Secure regression on distributed databases,” J. Comput. Graph. Stat., vol. 14, no. 2, 2005.
[7] 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, 2004, vol. 233.
[8] K. El Emam, S. Samet, L. Arbuckle, R. Tamblyn, C. Earle, and M. Kantarcioglu, “A secure distributed logistic regression protocol for the detection of rare adverse drug events,” J. Am. Med. Inform. Assoc. JAMIA, vol. 20, no. 3, pp. 453–461, May 2013.
[9] R. Hall, S. E. Fienberg, and Y. Nardi, “Secure multiple linear regression based on homomorphic encryption,” J. Off. Stat., vol. 27, no. 4, p. 669, 2011.
[10] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, “Privacy-Preserving Ridge Regression on Hundreds of Millions of Records,” 2012.
[11] F. Dankar, R. Brien, C. Adams, and S. Matwin, “Secure Multi-Party linear Regression,” in EDBT/ICDT Workshops, 2014, pp. 406–414.
[12] D. C. Montgomery, E. A. Peck, and G. G. Vining, Introduction to linear regression analysis, vol. 821. Wiley, 2012.
[13] G. A. Seber and A. J. Lee, Linear regression analysis, vol. 936. John Wiley & Sons, 2012.
[14] S. Paterlini and T. Minerva, “Regression model selection using genetic algorithms,” in Proceedings of the 11th WSEAS International Conference on RECENT Advances in Neural Networks, Fuzzy Systems & Evolutionary Computing, 2010, pp. 19–27.

[15]V. Ciriani, S. D. C. di Vimercati, S. Foresti, and P. Samarati, "κ-anonymity," in Secure Data Management in Decentralized Systems, Springer, 2007, pp. 323–353.
[16]G. Jagannathan, K. Pillaipakkamnatt, and D. Umano, "A secure clustering algorithm for distributed data streams," in Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on, 2007, pp. 705–710.
[17]P. K. Prasad and C. P. Rangan, "Privacy preserving BIRCH algorithm for clustering over arbitrarily partitioned databases," in Advanced Data Mining and Applications, Springer, 2007, pp. 146–157.
[18]J. Sakuma and S. Kobayashi, "Large-scale k-means clustering with user-centric privacy preservation," in Advances in Knowledge Discovery and Data Mining, Springer, 2008, pp. 320–332.
[19]K. El Emam, S. Samet, J. Hu, L. Peyton, C. Earle, G. C. Jayaraman, T. Wong, M. Kantarcioglu, F. Dankar, and A. Essex, "A Protocol for the secure linking of registries for HPV surveillance," PloS One, vol. 7, no. 7, p. e39915, 2012.
[20]A. T. Soodejani, M. A. Hadavi, and R. Jalili, "k-anonymity-based horizontal fragmentation to preserve privacy in data outsourcing," in Data and Applications Security and Privacy XXVI, Springer, 2012, pp. 263–273.
[21]S. Han and W. K. Ng, "Privacy-preserving linear fisher discriminant analysis," in Advances in Knowledge Discovery and Data Mining, Springer, 2008, pp. 136–147.
[22]P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in cryptology—EUROCRYPT'99, 1999, pp. 223–238.
[23]C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft, "Efficient rsa key generation and threshold paillier in the two-party setting," in Topics in Cryptology–CT-RSA 2012, Springer, 2012, pp. 313–331.
[24]Y. Desmedt, "Threshold cryptosystems," in Advances in Cryptology—AUSCRYPT'92, 1993, pp. 1–14.
[25]T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in Information Security Applications, Springer, 2011, pp. 44–60.
[26]H. Cohen, A Course in Computational Algebraic Number Theory. Springer-Verlag, 1993.