Privacy Preserving Regression Algorithms

Artak Amirbekyan, Vladimir Estivill-Castro

School of ICT, Griffith University,
Meadowbrook QLD 4131, Australia

Abstract. Regression is arguably the most applied data analysis method. Today there are many scenarios where data for attributes that correspond to predictor variables and the response variable itself are distributed among several parties that do not trust each other. Privacy-preserving data mining has grown rapidly studying the scenarios where data is vertically partitioned. While algorithms have been developed for many tasks (like clustering, association-rule mining and classification), for regression, the case of only two parties remains open. Also open is the most interesting case when the response variable is to be kept private. This paper provides the first set of algorithms that solves these cases. Our algorithms are practical and only require a commodity server (a supplier of random values) that does not collude with the parties. Our protocols are secure in the spirit of the the semi-honest model.

1 Introduction

Regression in one of the most widely used statistical tools in data analysis. Regression allows to model a relationship between an attribute considered the response and other attributes considered to influence such response. In this case, this is often called multiple regression. Linear regression assumes that the relationship between the response and a number of attributes is linear. Linear regression is perhaps the most successful statistical method in data analysis.

Statistical regression modeling typically assumes that the data is freely available. However, data collection may be among several agencies resulting in large and rich but distributed databases. The parties may not wish to share their data, or may be in no position to share their data although they would all benefit from the statistical analysis on the joined database. For example, data analysis is regarded as one of the most useful tools for the fight on crime [11]. However, the information needed resides with many different governments and/or corporations and these parties may mutually do not trust each other. Legal or constitutional limitations, or conflicts of interest may pose demands on privacy. But all parties are aware of the benefits brought by analyzing the collective datasets.

When privacy-preserving is an issue, integration of a single database is only a partial answer [5,8,9]. All partners of the collaboration promise to provide their private data, but none wants partners or any third party to learn their private data. This context demands effective and efficient algorithms for privacy preserving regression [8,9,10,15]. In the privacy preserving model, data is distributed over several parties and the goal is to compute linear regression by preserving the confidentiality of each party's data.

We present new methods for privacy-preserving regression analysis. Specifically, we provide advances in three aspects. First, our methods are the first methods to address the issue when the response variable is not common knowledge to all parties. Second, we also deal effectively with the case of only two parties. Third, we can produce coefficients and other intermediate data *in shares*, which allows repetition, re-calculation and detailed analysis of model quality and fit.

We will briefly introduce necessary terminology and discuss the current state of progress on privacy preserving regression in the next section. We then treat separately bilinear from multiple linear regression, because they require different solutions. We show then how to compute residuals which allows model evaluation and model selection.

2 Regression in the Privacy Preserving Context

Regression analysis examines the relationship of a dependent variable Y (the *response variable*) to specified independent variables (the *predictors*). Regression is a fundamental tool in data analysis used for prediction, modeling of casual relationships, and scientific hypothesis testing about relationships between variables among many other uses. The mathematical model of the relationship is the regression equation. *Linear regression* is a regression method that explores a linear relationship between the dependant variable Y and the k independent variables x_i (allowing an error term ϵ). That is, the form of the model (the regression equation) is $Y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \epsilon$. Linear regression is the most studied regression approach. The coefficients $\alpha = \beta_0$ and β_i are the parameters learned from the data.

In this paper, we study collaboration between several parties that wish to compute a regression equation using their collective data. However, each wants the others to find as little as possible of their own private data. We focus on vertically partitioned data (see Fig. 1). Every record in the database is an attribute-value vector[1]. One part of that vector is owned by Alice and the other part by Bob. In the case of more than two parties, then every party will own some part (a number of attributes) from the attribute-value vector. Note that, for vertically partitioned data, the more parties are involved, the more attributes are involved and the higher the

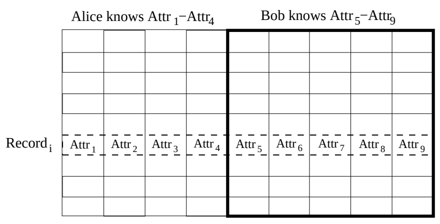

Fig. 1. Vertically partitioned data, Alice knows 4 attributes while Bob knows 5.

[1] For most data analysis and data mining algorithms, the data is encoded as vectors in high dimensional space. Attribute-vectors are the common input for learning algorithms like decision trees, artificial neural network or for clustering algorithms like k-Means or DBSCAN.

dimensions of the attribute-vectors. A direct use of regression algorithms on the union of the data requires one party to receive data (collect all attributes for all records) from all other parties, or all parties to send their data to a trusted central place. The recipient of the data would conduct the computation in the resulting union. Although there has been some work on privacy preserving integration [9,5], in settings where each party must keep their data private, this is usually unsatisfactory.

Also, for simplicity, we identify each predictor variable x_i with one party (so the dimension k of the records is also the number m of parties). Typically there will be less parties than dimensions (as in Fig. 1 where two parties have data for 9-dimensional records). However, we consider Alice as 4 virtual parties (one for each of the columns) and Bob as 5 virtual parties each controlling one of Bob's column. This simplifies the notation in the algorithms (and communication between two virtual parties of the same party just does not need to occur). It also may be the case that a predictor variable x_i is in fact a function of several attributes owned by one party; however, this again is a matter internal to that party and we will not address it further.

In order to have privacy preserving liner regression one must perform several steps which includes secure calculation of the regression coefficients β_i and model diagnostics. The calculation of the regression coefficients is an important step in regression but the diagnostics and model selection are even more important and challenging. Diagnostics checks whether a model is proper and the best possible or it needs revision by further analysis. This can be carried out by graphical tools that include plotting the *residual versus the predicted response* and/or *residual versus predictor* plots. Model selection can be performed iteratively, controlled by the analyst based on diagnostics analysis, or an automatically by stepwise regression, or exhaustively, that is, running over all possible models relying on some model selection criteria such as Mallow's C_p statistic.

The first solution for linear regression in the privacy context [3] was based on a series of protocols for matrix multiplication that were secure in a weak sense [15]. Other solutions addressed the simpler case of horizontally partitioned data [9]. Using Powell's algorithm for solving quadratic minimization an alternative was provided for the vertical partitioning case [12]. All of these assume that the response variable Y is known to all the parties. This reduces the cases where two different parties may attempt to understand the relationship between attributes in their data by means of linear regression. Specially, if there is no commonly known attribute that can act as the commonly known response. For example, one company may hold salary and education level for a large set of employees, while the medical insurer may hold data about the frequency of medical checkups. It would be difficult to explore the relationship between education level and the monitoring individuals perform on their health. Similarly, the potential relationships between types of treatment and professional activity (that could lead to patterns in some certain conditions due to the nature of the job). Similarly, if there was a third party, say a retailer, then it may be interesting to explore the types of expenses in relation to salary. This may enable the retailer to target its advertising, and the employer to offer some employment benefits according to some rank within the organization.

Although the previous solutions [3,12] provide privacy preserving calculation of regression coefficients for two parties holding **more that one attribute**, Vaidya and Clifton have remarked [15] on a potential privacy breach of the attribute values when using the residual versus predictor plots to determine whether the fitted model is proper. For example, Alice can generate the residual versus x_1 plot. In the plot, the coordinates of the points are exactly the values of x_1. If the plot is revealed to the other party (Bob), Bob may use the plot to recover accurate values of x_1 which are held by Alice. We will show that this situation can be avoided if Alice and Bob distribute residuals *in shares* and apply secure multi-party protocols for diagnosis analysis. But none of the solutions in the literature provide coefficients as well as residuals with shares.

Another privacy risk that was highlighted previously [15], is the case of only two parties each holding only one attribute. In this scenario, the disclosure of the residuals immediately results in the disclosure of the attribute values of the opposite party. We will also provide the first solution to this case.

3 Privacy Preserving Computation

We present several tools for privacy preserving computation. These tools were originally developed under the name of "secure multi-party computation" (SMC) [6]. Here Alice holds one input vector $\boldsymbol{x}$ and Bob holds an input vector $\boldsymbol{y}$. They both want to compute a function $f(\boldsymbol{x}, \boldsymbol{y})$ without each other learning anything about each other's input expect what can be inferred from $f(\boldsymbol{x}, \boldsymbol{y})$. Yao's Millionaires Problem [16] provides the origin for SMC. In the Millionaires, Alice holds a number a while Bob holds b. They want to identify who holds the larger value (they compute if $a > b$) without neither learning anything else about the others value. The function $f(x, y)$ is the predicate $f(x, y) = (x > y)$.

It is common to use a SMC-protocol as a sub-protocol of another that performs a more elaborate computation. In this case, it is not satisfactory for the parties to learn the output of the sub-protocol as this may lead to learning information about the inputs for the overall process. We are interested in covering the output of some particular SMC sub-protocol, and this is usually achieved by distributing the intermediate result among the parties in what is called *shares*. For example, we may be interested in finding the largest value among several parties. Using Yao's comparison protocol, we could compute which of two numbers are grater and use this information in the maximum-finding protocol. However, if the output of every comparison becomes public, then the parties would learn information beyond the maximum value, perhaps even which party holds the second largest value. It is much better to keep the results of individual comparisons as shares which on a second phase are summed up and find the maximum without no party learning the outcome of any comparison [1].

The origins of secure multi-party computation were in fact presented with the idea of each party receiving shares of the output, and this idea lead to a theoretical result that any function f with polynomial complexity could be described as a digital circuit of polynomial size where the parties could each be assigned shares of each logical gate. This theoretical result implies that any polynomial algorithm can be adapted for privacy preservation, under the semi-honest model of computation [2]. However, this result is in truly theoretical. We rarely can describe an algorithm, like matrix multiplication, as a large digital circuit, and such polynomial size circuit would be extremely large for the size of databases we have in mind in data analysis or data mining, and in particular for multiple regression.

One advantage of the "shares" theoretical result is that one can easily decompose the result $f(\boldsymbol{x}, \boldsymbol{y})$ into a share s_A for Alice and a share s_B for Bob, so that $s_A + s_B = f(\boldsymbol{x}, \boldsymbol{y})$, and use this as a sub-protocol in another more elaborate protocol, while neither party can find $f(\boldsymbol{x}, \boldsymbol{y})$ from their share. This usage of a protocol as a subroutine in another protocol enables construction of more complex and secure protocols, but transmits the impracticality of the generic "shares" further. We believe this has been somewhat over-used in the privacy preserving literature. There are several algorithms [7,13,14] that invoke a subroutine for Yao's comparison with shares, and all of them rely on the circuit evaluation generic "shares" theoretical solution by Goldreich [6]. Hence, they seem hard for implementation.

To produce algorithms for multiple regression, we will require some other secure multi-party computation sub-protocols, some of which are already in the literature and some we present here. We introduce a protocol to compute a division, when the parties themselves have shares of the divisor and the dividend. The result will be shared. The current SMC division protocol [2], does not provide an answer with shares, it gives an answer to one party only.

In the division protocol, Alice holds (a_1, a_2) and Bob holds (b_1, b_2), the goal is for Alice to obtain a value A and for Bob to receive a value B, where $A + B = (a_1 + b_1)/(a_2 + b_2)$. Thus, they share the outcome and also, Alice does not learn any of the values hold by Bob, while Bob does not discover any of the values hold by Alice. The steps of the protocol are as follows.

1. Alice produces a random number r_1 and Bob produces a random number r_2.
2. Using the so called scalar product protocol [2], Alice can obtain $r_2(a_2+b_2) = (a_2, 1)^T \cdot \begin{pmatrix} r_2 \\ r_2 b_2 \end{pmatrix}$, where Alice supplies $(a_2, 1)^T$ and Bob supplies $r_2 \begin{pmatrix} 1 \\ b_2 \end{pmatrix}$. It is important that the scalar product protocol in the literature [2] gives the answer to only one party, in this case Alice. Similarly, using the same protocol, but now in a way that only Bob learns the answer, and with inputs $r_1 \begin{pmatrix} 1 \\ a_2 \end{pmatrix}$ for Alice and $(b_2, 1)^T$ for Bob, we allow Bob to get $r_1(a_2 + b_2) = (b_2, 1)^T \cdot \begin{pmatrix} r_1 \\ r_1 a_2 \end{pmatrix}$.

[2] Secure multi-party computation under the semi-honest model [6] means all parties will follow the protocol since all are interested in the results. However, all parties can use all the information collected during the protocol to attempt to discover the private data or some private values from another party.

3. Alice and Bob again perform the scalar product [4], but a variant that provides an answer with shares. The inputs will be $\left(r_1 a_1, \frac{1}{r_2(a_2+b_2)}\right)^T$ for Alice and $\begin{pmatrix} \frac{1}{r_1(a_2+b_2)} \\ r_2 b_1 \end{pmatrix}$ for Bob. Thus Alice would obtain a value A and Bob would obtain a value B with the property that

$$A + B = \left(r_1 a_1, \frac{1}{r_2(a_2+b_2)}\right)^T \cdot \begin{pmatrix} \frac{1}{r_1(a_2+b_2)} \\ r_2 b_1 \end{pmatrix} = \frac{a_1+b_1}{a_2+b_2}.$$

4 Privacy Preserving Bivariate Linear Regression

Bivariate linear regression models the response variable Y as a linear function of just one predictor variable X; that is $Y = \alpha + \beta X + \epsilon$, where α and β are regression coefficients specifying the Y-intercept and slope of the line, respectively. These coefficients can be found by minimization of the error ϵ between the actual data and the estimate of the line. Given n sample data points of the form (a_1, b_1), $\cdots$, (a_n, b_n), then the regression coefficients estimated by the method of least squares are

$$\beta = \frac{\sum_{i=1}^n (a_i - \bar{a})(b_i - \bar{b})}{\sum_{i=1}^n (a_i - \bar{a})^2} \quad (1) \qquad \text{and} \qquad \alpha = \bar{b} - \beta \bar{a} \quad (2)$$

where $\bar{a}$ is the average of $a_1, \cdots, a_n$ and $\bar{b}$ is the average of $b_1, \cdots, b_n$.

When data is vertically partitioned, Alice would know all $a_1, \cdots, a_n$ and Bob will have $b_1, \cdots, b_n$. Thus, Alice and Bob can calculate each $\bar{a}$ and $\bar{b}$ without any communication. The goal would be for Alice and Bob to obtain the coefficients for $Y = \alpha + \beta X$, while they do not learn each other's data points. Note, however, that knowledge of β and α by Alice and Bob implies (because $\alpha = \bar{b} - \beta \bar{a}$, that each will learn something about each other's data. Alice will discover $\bar{b}$ and Bob $\bar{a}$. It is commonly accepted in secure multi-party computation that anything that can be learned from the output $f(\boldsymbol{x}, \boldsymbol{y})$, about the others party data is acceptable. We will present this situation first. The alternative, is that the output $f(\boldsymbol{x}, \boldsymbol{y})$ is in shares. We present this case second.

In order for the parties to find β, we provide the following protocol. First note that the dividend in Eq. (1) is a scalar product of two vectors, each know to one party only.

$$\sum_{i=1}^n (a_i - \bar{a})(b_i - \bar{b}) = \left((a_1 - \bar{a}), \cdots, (a_n - \bar{a})\right)^T \cdot \left((b_1 - \bar{b}), \cdots, (b_n - \bar{b})\right). \qquad (3)$$

Using scalar product protocol [2] which provides an answer to Alice only, she can then divide by $\sum_{i=1}^n (a_i - \bar{a})^2$ (which she owns) and obtain β. Alice would then pass β to Bob. This way Alice and Bob learn the coefficients with a protocol that is best possible in the sense that the protocol reveals to each party the final result and only what can be discovered using the final result and one's input.

However, if the coefficients are to be learned in shares $\alpha = s_a(\alpha) + s_b(\alpha)$ and $\beta = s_a(\beta) + s_b(\beta)$ (with $s_a(\alpha), s_a(\beta)$ known only to Alice and $s_b(\alpha), s_b(\beta)$

known only to Bob) we need additional care. By Eq. (2), if $\beta = s_a(\beta) + s_b(\beta)$, then $\alpha = \bar{b} + [s_a(\beta) + s_b(\beta)]\bar{a} = \bar{b} + s_b(\beta)]\bar{a} + s_a(\beta)\bar{a} = \left(1, \bar{a}, \bar{a}s_a(\beta)\right)^T \cdot \begin{pmatrix} \bar{b} \\ s_b(\beta) \\ 1 \end{pmatrix}$. Because the first vector above is known only to Alice and the second is known only to Bob, using the scalar product protocol [4] with shares would provide the required $s_a(\alpha)$ for Alice and $s_b(\alpha)$ for Bob with $\alpha = s_a(\alpha) + s_b(\alpha)$. Thus, providing the coefficients with shares reduces to providing β with shares.

We accomplish this requirement as follows. Recall that β is an expression (Eq. (1)) whose dividend is a scalar product (Eq. (3)). Thus, we can obtain the dividend as two values A_1 and B_1, with A_1 only known to Alice and B_1 only known to Bob. Let Alice generate a random number R, that she passes to Bob, and consider the following derivation.

$$\beta = \frac{A_1 + B_1}{(\sum_{i=1}^{n}(a_i - \bar{a})^2 + R) - R} = \frac{A_1 + B_1}{A_2 + B_2}.$$

This has the form of the division protocol with A_1, A_2 only known to Alice, and although B_1 is only known to Bob, $B_2 = R$ is known to both Bob and Alice. Nevertheless, we can apply the secure division protocol from Section 3. Knowledge of $B_2 = R$ by Alice results in Alice learning r_2, but interestingly enough, this is still insufficient for Alice to learn b_1 or Bob's share in the output [3]. This gives then the required shares for β.

We have provided two protocols. Firstly, Alice and Bob learn the coefficients α and β. In the second one, they learn these coefficients, but in shares. Both protocols are ideal, in the sense of privacy from the semi-honest model in SMC, as what each party learns about the others data is nothing more that what can be inferred from the specified output of the protocol and its own data.

5 Privacy Preserving Multiple Regression

In this section we investigate multiple regression. Here, several parties are involved with several attributes and the goal is again to obtain linear regression coefficient across different attributes. We once more do not assume a response variable in common. In fact, this enables cases where the response variable is an attribute known by one of the involved parties. This provides the ability to find relationships between different attributes of different parties. The only assumption now is that there are three or more non-virtual parties. ($m \geq 3$).

Using the least squares method, the vector of coefficients $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_m)$ is $\boldsymbol{\beta} = (X^T X)^{-1} X^T \boldsymbol{Y}$, where the matrix $X = (\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_m)$ has column vectors $\boldsymbol{X}_j$ and each $\boldsymbol{X}_j$ is owned by $j - th$ party. The response variable is a vector $\boldsymbol{Y}$ owned by one party only. The task here is to compute β without revealing any party's' data. We first present a protocol that reveals $\boldsymbol{\beta}$ to all parties. Let us first describe how to compute $X^T X$. We start with the three party case, ($m = 3$). If we have n data points of the form (a_i, b_i, c_i) with a_i known to Alice

[3] Full proof of this requires description of the scalar product in shares, because of space limitations this is included in an appendix for referees past the 12 page limit.

only, b_i known to Bob only and c_i known to Charles only, we have the following data matrix X given by

$$X^T = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \\ c_1 & c_2 & c_3 & \dots & c_n \end{pmatrix}.$$

Then, by using scalar product protocol [4], with output distributed in shares, whenever we have an matrix entry with data vectors belonging to two different parties, we obtain the following derivation[4] for the symmetric matrix X^TX.

$$X^TX = \begin{pmatrix} \boldsymbol{a}^T \cdot \boldsymbol{a} & \boldsymbol{a}^T \cdot \boldsymbol{b} & \boldsymbol{a}^T \cdot \boldsymbol{c} \\ \boldsymbol{b}^T \cdot \boldsymbol{a} & \boldsymbol{b}^T \cdot \boldsymbol{b} & \boldsymbol{b}^T \cdot \boldsymbol{c} \\ \boldsymbol{c}^T \cdot \boldsymbol{a} & \boldsymbol{c}^T \cdot \boldsymbol{b} & \boldsymbol{c}^T \cdot \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}^T \cdot \boldsymbol{a} & V^A_{ab} + V^B_{ab} & V^A_{ac} + V^C_{ac} \\ V^A_{ab} + V^B_{ab} & \boldsymbol{b}^T \cdot \boldsymbol{b} & V^B_{bc} + V^C_{bc} \\ V^A_{ac} + V^C_{ac} & V^B_{bc} + V^C_{bc} & \boldsymbol{c}^T \cdot \boldsymbol{c} \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{a}^T \cdot \boldsymbol{a} & V^A_{ab} & V^A_{ac} \\ V^A_{ab} & 0 & 0 \\ V^A_{ac} & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & V^B_{ab} & 0 \\ V^B_{ab} & \boldsymbol{b}^T \cdot \boldsymbol{b} & V^B_{bc} \\ 0 & V^B_{bc} & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & V^C_{ac} \\ 0 & 0 & V^C_{bc} \\ V^C_{ac} & V^C_{bc} & \boldsymbol{c}^T \cdot \boldsymbol{c} \end{pmatrix}$$

Thus, X^TX is computed in our protocol by matrix addition (also called secure sum [15]) where each party owns one matrix. In order to add them securely, Alice (the first party) cat generate a random matrix and pass it to Charles (the third party) who will add his matrix to the random matrix received from Alice. Charles will send this sum to Bob. Next, Bob will add his matrix to the matrix received from Charles and send to Alice. Alice subtracts the original random matrix and add her matrix to obtain X^TX.

For the case $m \geq 3$ (with n data points), we use the scalar product with shares to obtain $\boldsymbol{p}_i^T \cdot \boldsymbol{p}_j$ as $V^i_{p^ip^j} + V^j_{p^ip^j}$ when $\boldsymbol{p}_i, V^i_{p^ip^j}$ is known only to the i-th party and $\boldsymbol{p}_j, V^j_{p^ip^j}$ is known only to the j-th party, whenever $i \neq j$. Thus,

$$X^TX = \begin{pmatrix} p^1_1 & \cdots & p^1_n \\ p^2_1 & \cdots & p^2_n \\ \vdots & \ddots & \vdots \\ p^m_1 & \cdots & p^m_n \end{pmatrix} \cdot \begin{pmatrix} p^1_1 & \cdots & p^m_1 \\ p^1_2 & \cdots & p^m_2 \\ \vdots & \ddots & \vdots \\ p^1_n & \cdots & p^m_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}_1^T \cdot \boldsymbol{p}_1 & \boldsymbol{p}_1^T \cdot \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_1^T \cdot \boldsymbol{p}_m \\ \boldsymbol{p}_2^T \boldsymbol{p}_1 & \boldsymbol{p}_2^T \cdot \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_2^T \cdot \boldsymbol{p}_m \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{p}_m^T \cdot \boldsymbol{p}_1 & \boldsymbol{p}_m^T \cdot \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_m^T \cdot \boldsymbol{p}_m \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{p}_1^T \cdot \boldsymbol{p}_1 & V^1_{p^1p^2} + V^2_{p^1p^2} & \cdots & V^1_{p^1p^m} + V^m_{p^1p^m} \\ V^1_{p^2p^1} + V^2_{p^2p^1} & \boldsymbol{p}_2^T \cdot \boldsymbol{p}_2 & \cdots & V^2_{p^2p^m} + V^m_{p^2p^m} \\ \vdots & & \ddots & \vdots \\ V^1_{p^mp^1} + V^m_{p^mp^1} & V^2_{p^mp^2} + V^m_{p^mp^2} & \cdots & \boldsymbol{p}_m^T \cdot \boldsymbol{p}_m \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{p}_1^T \cdot \boldsymbol{p}_1 & V^1_{p^1p^2} & \cdots & V^1_{p^1p^m} \\ V^1_{p^2p^1} & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ V^1_{p^mp^1} & 0 & \cdots & 0 \end{pmatrix} + \begin{pmatrix} 0 & V^2_{p^1p^2} & \cdots & 0 \\ V^2_{p^2p^1} & \boldsymbol{p}_2^T \cdot \boldsymbol{p}_2 & \cdots & V^2_{p^2p^m} \\ \vdots & \ddots & & \vdots \\ 0 & V^2_{p^2p^m} & \cdots & 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 & \cdots & V^m_{p^1p^m} \\ 0 & \cdots & V^m_{p^2p^m} \\ \vdots & \ddots & \vdots \\ V^m_{p^1p^m} & \cdots & \boldsymbol{p}_m^T \cdot \boldsymbol{p}_m \end{pmatrix}.$$

We apply a similar strategy in our protocol for the computation of $X^T\boldsymbol{Y}$.

$$X^T\boldsymbol{Y} = \begin{pmatrix} p^1_1 & \cdots & p^1_n \\ p^2_1 & \cdots & p^2_n \\ \vdots & \ddots & \vdots \\ p^m_1 & \cdots & p^m_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{p}_1^T \cdot \boldsymbol{Y} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \boldsymbol{p}_2^T \cdot \boldsymbol{Y} \\ \vdots \\ 0 \end{pmatrix} + \ldots + \begin{pmatrix} 0 \\ \\ \vdots \\ \boldsymbol{p}_m^T \cdot \boldsymbol{Y} \end{pmatrix}.$$

[4] The super-index provides the owner party.

We have that each party knows a vector and their sum is $X^T\boldsymbol{Y}$. Thus, Alice generates a vector $\boldsymbol{r}$ with n different random values, passes this $\boldsymbol{r}$ to the last party. Each party adds the vector given to its own vector and passes to the previously numbered party. When the vector is back to Alice, she subtracts $\boldsymbol{r}$ and adds her vector to obtain $X^T\boldsymbol{Y}$. Note that Alice does not know X^T; thus, knowledge of $X^T\boldsymbol{Y}$ will not enable it to derive the private values of $\boldsymbol{Y}$ when Alice is not the party supplying $\boldsymbol{Y}$ (even knowledge of $\boldsymbol{p}_1^T$ and $\boldsymbol{p}_1^T \cdot \boldsymbol{Y}$ does not reveal anything (unless $n = 1$, but we usually have more than one data point). Also, if $\boldsymbol{Y}$ is known by Alice and no other party, Alice cannot learn data from another party.

Hence Alice will get X^TX and $X^T\boldsymbol{Y}$. She can compute $\boldsymbol{\beta}$ by inverting X^TX and multiplying with $X^T\boldsymbol{Y}$. In the last step of the protocol, Alice broadcasts $\boldsymbol{\beta}$ to all parties.

We now introduce a protocol that distributes $\boldsymbol{\beta}$ in shares to at least two parties. This protocol works as before except that now.

1. All parties will engage in the protocol for calculating X^TX and the output will go to Alice.
2. All parties and the party holding $\boldsymbol{Y}$ (say Yuri) will compute $X^T\boldsymbol{Y}$ and the output will go to a party different than Alice. The easiest is for $X^T\boldsymbol{Y}$ to go to Yuri.
3. Alice and Yuri will multiply the matrix $A = (X^TX)^{-1}$ and the vector $\boldsymbol{B} = X^T\boldsymbol{Y}$ to obtain shares.

Step 1 and Step 2 are essentially as before. Step 3 can be archived again by using the scalar product protocol [4] that splits into shares.

$$A\boldsymbol{B} = \begin{pmatrix} (a_1^1, \cdots, a_m^1)^T \cdot \boldsymbol{B} \\ (a_1^2, \cdots, a_m^2)^T \cdot \boldsymbol{B} \\ \vdots \\ (a_1^m, \cdots, a_m^m)^T \cdot \boldsymbol{B} \end{pmatrix} = \begin{pmatrix} V^1_{\boldsymbol{a^1 b}} + V^2_{\boldsymbol{a^1 b}} \\ V^1_{\boldsymbol{a^2 b}} + V^2_{\boldsymbol{a^2 b}} \\ \vdots \\ V^1_{\boldsymbol{a^m b}} + V^2_{\boldsymbol{a^m b}} \end{pmatrix} = \begin{pmatrix} V^1_{\boldsymbol{a^1 b}} \\ V^1_{\boldsymbol{a^2 b}} \\ \vdots \\ V^1_{\boldsymbol{a^m b}} \end{pmatrix} + \begin{pmatrix} V^2_{\boldsymbol{a^1 b}} \\ V^2_{\boldsymbol{a^2 b}} \\ \vdots \\ V^2_{\boldsymbol{a^m b}} \end{pmatrix} \qquad (4)$$

In this way, the output $\boldsymbol{\beta}$ will be shared between two parties. If the fact that X^TX is known to Alice is of some concern [10], a variant of the protocol where X^TX is discovered in distributed shares can be obtained if Alice and Bob skip the last matrix transmission in the protocol from the previous subsection. That is, Bob does not send his sum matrix to Alice. Thus, Alice will hold $A - R$ and Bob will hold $B + C + R$, which will serve as shares for the output. Similarly, rather than Yuri holding $X^T\boldsymbol{Y}$, the computation of the sum by passing a vector and accumulating can be halted before the last transmission. Then $(X^TX)^1X^T\boldsymbol{Y} = (A_1 + B_1)^{-1}(\boldsymbol{Z}_1 + \boldsymbol{Z}_2) = (A_1 + B_1)^{-1}\boldsymbol{Z}_1 + (A_1 + B_1)^{-1}\boldsymbol{Z}_2$. Multiplication and inversion of a matrix sum can be performed with dedicated protocols [3].

6 Model diagnosis

As we mentioned earlier, the calculation of $\boldsymbol{\beta}$ is an important step in regression, but it is only the first step. The other steps include diagnostics and model selection. Statistics reflecting the goodness of fit of a model include the correlation coefficient R^2 and the adjusted R^2. The residuals play an essential role in diagnostics. Once $\boldsymbol{\beta}$ is available, we can calculate the fitted or predicted responses as $\hat{\boldsymbol{Y}} = X\boldsymbol{\beta}$. The column vector of residues is $\hat{\varepsilon} = \boldsymbol{Y} - \hat{\boldsymbol{Y}}$ and the residual for the $i - th$ data point is $\hat{\varepsilon}_i = y_i - \hat{y}_i$. Then

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}. \tag{5}$$

For simplicity, we will assume that only 3 parties are involved. For more parties, we only need to extend the calculation of a sum of vectors among more parties by passing around an accumulator vector. If $\boldsymbol{\beta}$ has been calculated to make it available to all parties (the version without shares), then the data matrix X has columns owned by each party and

$$X\boldsymbol{\beta} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & & \\ a_n & b_n & c_n \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} a_1\beta_1 \\ a_2\beta_1 \\ \vdots \\ a_n\beta_1 \end{pmatrix} + \begin{pmatrix} b_1\beta_2 \\ b_2\beta_2 \\ \vdots \\ b_n\beta_2 \end{pmatrix} + \begin{pmatrix} c_1\beta_3 \\ c_2\beta_3 \\ \vdots \\ c_n\beta_3 \end{pmatrix}. \tag{6}$$

Again, this is a sum of vectors each known by one party and can be computed by summing and passing an accumulator initiated with random values by the first party.

When $\boldsymbol{\beta}$ is not publicly available, that is $\boldsymbol{\beta}$ is distributed by shares (Equation (4)), then privacy-preserving calculation is more challenging. Here we have

$$\begin{aligned} X\boldsymbol{\beta} &= \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & & \\ a_n & b_n & c_n \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & & \\ a_n & b_3 & c_3 \end{pmatrix} \cdot \left[\begin{pmatrix} V^1_{\boldsymbol{p^1}\boldsymbol{\beta}} \\ V^1_{\boldsymbol{p^2}\boldsymbol{\beta}} \\ V^1_{\boldsymbol{p^3}\boldsymbol{\beta}} \end{pmatrix} + \begin{pmatrix} V^2_{\boldsymbol{p^1}\boldsymbol{\beta}} \\ V^2_{\boldsymbol{p^2}\boldsymbol{\beta}} \\ V^2_{\boldsymbol{p^3}\boldsymbol{\beta}} \end{pmatrix} \right] \\ &= \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & & \\ a_n & b_n & c_n \end{pmatrix} \cdot \begin{pmatrix} V^1_{\boldsymbol{p^1}\boldsymbol{\beta}} \\ V^1_{\boldsymbol{p^2}\boldsymbol{\beta}} \\ V^1_{\boldsymbol{p^3}\boldsymbol{\beta}} \end{pmatrix} + \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & & \\ a_n & b_n & c_n \end{pmatrix} \cdot \begin{pmatrix} V^2_{\boldsymbol{p^1}\boldsymbol{\beta}} \\ V^2_{\boldsymbol{p^2}\boldsymbol{\beta}} \\ V^2_{\boldsymbol{p^3}\boldsymbol{\beta}} \end{pmatrix}. \end{aligned} \tag{7}$$

Hence, we need a protocol for securely computing $X\boldsymbol{V^1_{p\beta}}$ and $X\boldsymbol{V^2_{p\beta}}$ where X is the vertically partitioned matrix and $\boldsymbol{V^1_{p\beta}}$ is a vector belonging to one of the parties, in our case we can assume it is Alice. The $X\boldsymbol{V^2_{p\beta}}$ belongs to Yuri, which may or may not have any values in X. If we can calculate the scalar product when one vector is vertically partitioned and the other one has all its entries known to one party, the computation of $X\beta$ will again reduce to a sum of vectors distributed among the parties.

We are unaware of such protocol in the literature, so we propose here a solution based on scalar product protocol [4] that provides an answer in shares.

Protocol for computing $\boldsymbol{p}^T \cdot \boldsymbol{y}$, where $\boldsymbol{p} = (p_1, \ldots, p_m)$, each entry p_i is known to the i-th party and the vector $\boldsymbol{y}$ is know to the first party [5].

[5] The case where the owner of $\boldsymbol{y}$ is not an owner of an entry p_i can be handled by this same protocol, but has even more relaxed privacy settings.

1. The commodity server generates two random vectors [6] $\boldsymbol{\Psi}$ and $\boldsymbol{\Pi}$ of size m, and lets $r_a + r_b = \boldsymbol{\Psi}^T \cdot \boldsymbol{\Pi}$, where r_a (or r_b) is a randomly generated number. Then the server sends $(\boldsymbol{\Psi}, r_a)$ to the first party (lets call it Alice). It send r_b to the second party (say Bob). It also sends $\boldsymbol{\Pi}_i$ to the i-th party.
2. Alice computes a perturbed version $\hat{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\Psi}$ of its vector and sends the i-th entry to the i-th party. Each party computes $p_i\hat{y}_i = p_i y_i + p_i\boldsymbol{\Psi}_i$. That is, the i-th party gets $p_i(y_i + \boldsymbol{\Psi}_i)$.
3. Each of the parties perturbs its value with the random number provided by the commodity server $\hat{p}_i = p_i + \boldsymbol{\Pi}_i$ and sends it to Alice. Thus, Alice obtains the vector $\boldsymbol{p} + \boldsymbol{\Pi}$.
4. The parties engage in a circular accumulator sum, by which the first party passes $p_1 y_1 + p_1\boldsymbol{\Psi}_1$ to the m-th party. The i-th party adds $p_i y_i + p_i\boldsymbol{\Psi}_i$ to the sum and passes it to the $(i-1)$-th party until the second party (Bob) has $\boldsymbol{p}^T \cdot \boldsymbol{y} + \boldsymbol{p}^T \cdot \boldsymbol{\Psi}$.
5. Bob generates a random number V_2, and computes $\boldsymbol{p}^T \cdot \hat{\boldsymbol{y}} + (r_b - V_2)$. He sends this result to Alice.
6. Alice adds $r_a - \boldsymbol{\Psi}^T \cdot (\boldsymbol{p} + \boldsymbol{\Pi})$ to the value from Bob and calls it V_1. This is $V_1 = r_a - \boldsymbol{\Psi}^T \cdot (\boldsymbol{p} + \boldsymbol{\Pi}) + \boldsymbol{p}^T \cdot \hat{\boldsymbol{y}} + (r_b - V_2) = r_a + r_b - \boldsymbol{\Psi}^T \cdot \boldsymbol{\Pi} - \boldsymbol{\Psi}^T \cdot \boldsymbol{p} + \boldsymbol{p}^T \cdot (\boldsymbol{y} + \boldsymbol{\Psi}) - V_2 = \boldsymbol{p}^T \cdot \boldsymbol{y} - V_2$.

This protocol produces distributed shares V_1 for Alice and V_2 for Bob. The shares appear random to each but $V_1 + V_2 = \boldsymbol{p}^T \cdot \boldsymbol{y}$.

Thus, using this later scalar product protocol for the scalar product calculations in Equation (7), then every party will get its shares and

$$\hat{\boldsymbol{Y}} = X\beta = \begin{pmatrix} a_1 & b_1 & c_1 \\ \vdots & & \\ a_n & b_n & c_n \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} V_1^1 \\ \vdots \\ V_n^1 \end{pmatrix} + \begin{pmatrix} V_1^2 \\ \vdots \\ V_n^2 \end{pmatrix} + \begin{pmatrix} V_1^3 \\ \vdots \\ V_n^3 \end{pmatrix} + \begin{pmatrix} V_1^4 \\ \vdots \\ V_n^4 \end{pmatrix}. \quad (8)$$

It is clear now that using a secure accumulator sum protocol we can obtain the column vector of residues $\hat{\varepsilon} = \boldsymbol{Y} - \hat{\boldsymbol{Y}}$. Using this, the coefficient R^2 can also be computed by the parties without none revealing their data.

7 Conclusion

We have presented practical algorithms for performing privacy preserving regression in the more sensitive case, namely, where the response variable is private. Naturally, our methods apply as well when the response variable is public. Moreover, we have resolved both, the case where we have two parties and the general case of more than two parties. Most importantly, we have addressed the second phase of the regression task, the model valuation phase. This last point is very important, as a poor model fit may indicate the need to repeat the first phase. Preserving privacy while performing both phases several times is crucial for the overall success of the regression task. If there were information leaks in either phase, iteration of the phases would increase the lack of privacy.

[6] All entries are random numbers.

Privacy has a cost trade-off. Our algorithms are efficient because they offer only a constant overhead and are linear in the number of parties. The secure scalar product performs $4n$ computations rather than n for two vectors of dimension n. All our protocols are based on the same number of scalar product operations and occasionally an accumulator sum. This results in an overall complexity of $O(4mC(n))$ where $C(n)$ is the complexity on a consolidated database.

References

1. A. Amirbekyan and V. Estivill-Castro. The privacy of k-nn retrieval for horizontal partitioned data — new methods and applications. ADC-2007, vol 63 of *CRPIT*, 33–42, Ballarat, Australia, CORE.
2. W. Du and M.J. Atallah. Privacy-preserving cooperative statistical analysis. *17th ACSAC-2001*, 102–110, New Orleans, 10-14 ACM SIGSAC, IEEE Computer Soc.
3. W. Du, Y.-S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. *2004 SIAM ICDM*, Lak Buena Vista, Florida.
4. W. Du and Z. Zhan. Building decision tree classifier on private data. 1–8, 2002. IEEE ICDM Workshop Proceedings, Vol 14 CRPIT.
5. V. Estivill-Castro and A. Hajasien. Fast private association rule mining by a protocol securely sharing distributed data. *ISI-2007*, New Brunswick, New Jersey, IEEE Computer Soc. to appear.
6. O. Goldreich. *The Foundations of Cryptography*, vol 2, chapter General Cryptographic Protocols. Cambridge U. 2004.
7. M. Kantarcioğlu and C. Clifton. Privatly computing a distributed k-nn classifier. *8-th PKDD*, vol 3202, 279–290. Springer LNCS, 2004.
8. A. F. Karr, X. Lin, A. P. Sanil, and J.P. Reiter. Regression on distributed databases via secure multi-party computation. *DG.O*, 405–406, 2004.
9. A. F. Karr, X. Lin, A. P. Sanil, and J.P. Reiter. Secure regression on distributed databases. *J. Computational and Graphical Studies*, 14(2):1–18, 2005.
10. A. F. Karr, X. Lin, A. P. Sanil, and J.P. Reiter. Secure statistical analysis of distributed databases. *Statistical Methods in Counterterrorism*, 237–261. Springer, 2006.
11. J. Mena. *Investigative Data Mining for Security and Criminal Detection.* Butterworth-Heinemann, US, 2003.
12. A. P. Sanil, A. F. Karr, X. Lin, and J.P. Reiter. Privacy preserving regression modelling via distributed computation. , *KDD*, 677–682, Seattle, 2004. ACM.
13. M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. 2006 IEEE Int. Workshop on Privacy Aspects of Data Mining.
14. J. Vaidya and C. Clifton. Privacy-preserving outlier detection. *4th IEEE ICDM 2004*, Brighton, UK, IEEE Computer Soc..
15. J. Vaidya, C. Clifton, and M. Zhu. *Privacy Preserving Data Mining*, vol 19 of *Advances in Information Security.* Springer, New York, 2006.
16. A.C. Yao. Protocols for secure computation. *IEEE Symposium FOCS* , 160–164. IEEE Computer Soc., 1982.

Appendix for Interested Referees

Three models have been proposed for assessing the privacy of protocols in privacy-preserving data mining. The most common model is the *semi-honest* model, while less common are the *malicious model* and the *weak* model. The weak model was used for many algorithms involving matrix operations and in particular, linear regression [3,15]. In this model, security is regarded with respect to certainty. Therefore, one party is considered to not have bridged the security as long as there are an infinite number of possibilities for the values of the other parties. This model has been criticized, and in many cases rejected, because it can consider secure a protocol where one party learns significant information about another party's data. For example, Alice could learn that Bob's b_1 value is in a small range. While there are an infinite number of rationals (or reals) in this interval, this could provide enough precision for it to be considered a security leak. Learning or discovering an interval is discovering a distribution of the value. If the distribution has very small variance, although a large range, the security leak could be serious.

The malicious models expects both parties to behave as destructively as possible in attempting to discover data from one another. Thus, parties may supply false data, and interrupt the protocol. This model has been relegated to extreme cases, where the parties are not interested in the final result. Parties are not collaborating, but are attempting to infiltrate and perhaps damage each other. Thus, the semi-honest model has prevailed as the most common model. The formal definition of the semi-honest model is rather technical as it is the mechanisms to prove security of protocols. However, we show that demonstrating that a protocol is secure in the spirit of this protocol is not beyond a clear and transparent argument. One has to demonstrate that each party does not learn anything about another party's data except what can be learned from its own data and the result. All messages received appear as random values (as if they were generated by an oracle [6]), and thus, the party could complete the protocol correctly in polynomial time even if the messages were substituted by the random values provided by the oracle. We are to assume that all parties will behave in this way, and will follow and complete the protocol with genuine interest in the results, therefore, not supplying false data that would make the result invalid.

While some protocols for vector and matrix operations have been dismissed as only secure on the weak-model and not in the semi-honest model, we believe one cannot discard the merit of these protocols, specially if they are regarded as not secure in the semi-honest model by a technicality. Case in point is the protocol for scalar product that provides the output in shares [4]. This protocol is regarded as secure in the weak-model sense because it requires a commodity server. We argue here that the commodity server does not contradict the spirit of the semi-honest model. We can consider the commodity server as a third party in the protocol, with empty input and empty output. The three parties, then would be interested in computing $f(\boldsymbol{x}, \boldsymbol{y}, \lambda) = s_A + s_B$, where the input λ of the third party (the commodity server) is empty and will not affect the output value $s_A + s_b$ discovered by Alice and Bob with respected private shares. As long as the third party does not discover anything about Alice input $\boldsymbol{x}$, Bob's input $\boldsymbol{y}$r, Alice share s_A and Bob's share s_B, then the protocol is secure in the spirit of the semi-honest model. In fact, many times a protocol in the semi-honest model among more than two parties requires a sub-protocol in which two parties compute a value with the assistance of a third.

In what follows, we reproduce here the scalar product protocol from Du and Zhan [4] and show that neither party (including the commodity server) learns anything beyond its own input and what can be inferred from the result. Therefore, this protocol is secure in a sense stronger than the weak model and is secure in the spirit of the semi-honest model.

Protocol for computing $\boldsymbol{x}^T \cdot \boldsymbol{y}$, where $\boldsymbol{x}$ is known to Alice, and the vector $\boldsymbol{y}$ is know to Bob. The output is $V_1 + V_2 = \boldsymbol{x}^T \cdot \boldsymbol{y}$ so that V_1 is know to Alice and only Alice, while V_2 is known to Bob and only Bob. That is, they compute the dot product of two vectors in shares.

1. The commodity server generates two random vectors $\boldsymbol{\Psi}$ and $\boldsymbol{\Pi}$ of size n, and lets $r_a + r_b = \boldsymbol{\Psi}^T \cdot \boldsymbol{\Pi}$, where r_a (or r_b) is a randomly generated number. Then the server sends $(\boldsymbol{\Psi}, r_a)$ to the first party (Alice). It send $(\boldsymbol{\Pi}, r_b)$ to the second party (Bob).
2. Alice computes a perturbed version $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\Psi}$ of its vector and sends $\hat{\boldsymbol{x}}$ to Bob.
3. Bob also perturbs its vector with the random vector provided by the commodity server $\hat{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\Pi}$ and sends it to Alice. Thus, Alice obtains the vector $\boldsymbol{y} + \boldsymbol{\Pi}$.
4. Bob generates a random number V_2, and computes $\boldsymbol{y}^T \cdot \hat{\boldsymbol{x}} + (r_b - V_2)$. He sends this result to Alice.
5. Alice adds $r_a - \boldsymbol{\Psi}^T \cdot (\boldsymbol{y} + \boldsymbol{\Pi})$ to the value received from Bob and calls it V_1. This is $V_1 = r_a - \boldsymbol{\Psi}^T \cdot (\boldsymbol{y} + \boldsymbol{\Pi}) + \boldsymbol{y}^T \cdot \hat{\boldsymbol{x}} + (r_b - V_2) = r_a + r_b - \boldsymbol{\Psi}^T \cdot \boldsymbol{\Pi} - \boldsymbol{\Psi}^T \cdot \boldsymbol{y} + \boldsymbol{y}^T \cdot (\boldsymbol{x} + \boldsymbol{\Psi}) - V_2 = \boldsymbol{y}^T \cdot \boldsymbol{x} - V_2$.

Let us remark first that the above description enables to show that knowledge of $n-1$ entries on Bob's vector $\boldsymbol{y}$ by Alice is insufficient for Alice to learn the remaining entry. There will be still two unknowns (V_2 being one) in the equation $V_1 + V_2 = \boldsymbol{x}^T \cdot \boldsymbol{y}$ for Alice. Thus, the division protocol in Sec. 4 is secure.

The second point correspond to the fact that Bob, Alice and the commodity server do not learn anything about each other's data. Clearly the commodity server does not learn anything since it never receives any messages. Bob receives a perturbed vector from Alice, with all entries appearing random. Symmetrically, Alice receives Bob's vector completely masked by random values and thus these also appear as totally random values [7]. Bob generates its own random V_2 and will not receive anything else from Alice; therefore he cannot learn anything about Alice's data. Although Alice receives the additional value $\boldsymbol{y}^T \cdot \hat{\boldsymbol{x}} + (r_b - V_2)$ from Bob, $\boldsymbol{y}^T \cdot \hat{\boldsymbol{x}} + r_b$ is totally masked by the random value V_2. Alice cannot learn anything either.

[7] In practice, these values could range over a very large field F, and display a uniform distribution on such field; making impossible to either party even to learn their magnitude.