Secure Multi-Party linear Regression

Fida Dankar
University of Ottawa
IBM Canada
fidamark@ca.ibm.com

Renaud Brien
University of Ottawa
Rbrie047@uottawa.ca

Carlisle Adams
University of Ottawa
cadams@eecs.uottawa.ca

Stan Matwin
Dalhousie University
stan@cs.dal.ca

ABSTRACT
Increasing efficiency in hospitals is of particular importance. Studies that combine data from multiple hospitals/data holders can tremendously improve the statistical outcome and aid in identifying efficiency markers. However, combining data from multiple 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 privacy preserving linear regression model for the analysis of data coming from several sources. The protocol uses a semi-trusted third party and delivers on privacy and complexity.

Categories and Subject Descriptors
D.3.3 [**Computers and Society**]: Public Policy Issues– *Privacy*

H.2.8 [**Database Management**]: Database Applications– *Data Mining*.

General Terms
Algorithms, Security, Theory.

Keywords
Linear regression, privacy preserving data mining, secure multiparty computation.

1. INTRODUCTION
Hospitals are under a lot of pressure to increase their efficiency. They need to see more patients and reduce their costs without increasing resources [1]. Increasing the efficiency of critical resources, such as surgeons and operating rooms, while keeping the same quality of care is of particular focus [1]. For example, surgery completion time is an indicator of critical resources efficiency. Several studies have presented interesting explanation for the variation in surgery completion times [2]–[4], among the culprits are individual, team and organizational experience, learning curve heterogeneity and workload. These studies however work with raw data coming from a single source.

7th International Workshop on Privacy and Anonymity in the Information Society (PAIS'14) March 28, 2014, Athens, Greece

Combining data from multiple sources or hospitals is necessary to have high statistical power and sufficient heterogeneity among the subjects. However, combining data from multiple sources and performing statistical analysis on the union of the data poses privacy concerns [5].

A number of protocols have been proposed in the literature to address the privacy concerns; 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 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 involved sites, and the complexity for the Evaluator is linear in the number of sites. Private health information is preserved and the statistical outcome retains the same precision as that of raw data. Moreover, contrary to previous approaches, ours is complete. It not only calculates the linear regression parameters of a fixed model, but also includes model diagnostics and selection, which are the more important and challenging steps [5].

2. LINEAR REGRESSION
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). Fitting a linear regression model consists of a sequence of steps including estimation, diagnostics and model selection [10]. In what follows we give an overview of the steps involved, for more information the reader is referred to [10], [11]:

Assume that a dataset is composed of n input variables $x_i \in \Re^{|D|}$ ($\Re$ is the set of real numbers and D is the set of attributes), and n output variables $y_i \in \Re$. We denote by X the $n \times D$ input matrix $[x_i]$, and by Y the n output vector. 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 $f : \Re^{|d|} \rightarrow \Re$ that describe this dependency (of the ouput 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 $\beta = \begin{bmatrix} \beta_1 \\ : \\ \beta_d \end{bmatrix} \in \Re^{|d|}$, 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 $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ : \\ \beta_d \end{bmatrix}$, and to augment every row $x_i^{|d|}$ with 1 (so $x_i^{|d|}$ is set to $[1 \quad x_i^{|d|}]$), with that the formulae above can be restated as: $y_i \approx f(x_i) = \beta x_i^{|d|}$. 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 fit 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 used is referred to as the "least squares method" and it is equivalent to solving the following equation:

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

The process of determining the best subset $d \subseteq D$ is referred to as model selection. Model diagnostics are used to assess whether a fixed model (i.e. a regression model for a fixed d) is proper. One statistics that reflects the goodness of fit for a fixed model is the adjusted R^2 measure:

$$R_a^2 = 1 - \frac{(n-1)\sum_{i=1}^{n}(y_i - \hat{y}_i)}{(n-d-1)\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad (2)$$

Where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the output variable mean.

For a fixed model, once β is available, $\hat{y}_i = \beta x_i^{d}$ can be calculated followed by R_a^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. This is known as privacy-preserving 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 $X_k = \begin{bmatrix} x_{n_{k-1}+1} \\ : \\ x_{n_k} \end{bmatrix}$.

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 $X^T X$ by removing all entries ij from matrix $X^T X$ 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:

   $X^{d^T} X^d = \sum_{i=1}^{k} (X_i^{d^T} X_i^d)$ and that

   $X^{d^T} Y = \sum_{i=1}^{k} X_i^{d^T} Y_i$.

   Hence Equation (1) is equivalent to:

   $$\beta = (\sum_{i=1}^{k} (X_i^{d^T} X_i^d))^{-1} (\sum_{i=1}^{k} X_i^{d^T} Y_i) \qquad (3)$$

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 protocol, 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 R_a^2 for that given d.There are known iterative protocols for choosing the best subset d of independent variables [10], [11]. A common technique is to start with some basic set of attributes $d_0 \subseteq D$ and find its corresponding β and R_a^2. Additional attributes can then enter the analysis one by one and the effect of each can be studied separately through R_a^2. An outline of the algorithm is presented in Figure 1, 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], [11].

4. RELATED WORK

A number of protocols have been proposed for the collaborative computation of linear regression when data is horizontally distributed among the different parties. These protocols do not iterate over the attributes, they only offer fixed model solution (i.e. a solution for Equation (3)) and thus do not deal with model diagnostics.

The protocol in [7] suggests that the sites could share their local aggregate information. In other words site i would share with the other sites the aggregate values: ${X_i}^T X_i$ and ${X_i}^T Y_i$. This way, each site can add the aggregate information to obtain $X^T X$ and $X^T Y$, find the inverse of $X^T X$, then use Equation (1) to estimate the parameters of the regression. This method, although efficient, was criticized for being non-private as it shares local aggregate statistics [5], [8].

Another protocol due to Karr et al [6] suggests using secure multiparty computation in order to securely sum the local statistics: ${X_i}^T X_i$ and ${X_i}^T Y_i$, the final sum $X^T X$ and $X^T Y$ is then shared among the different sites. This protocol, although efficient, was also deemed to be non-private [8].

Three more protocols have been suggested to solve this problem, 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 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 their extensive use of the secure multiparty matrix multiplication protocol [12] extended to k parties. Each use of this k-party multiplication requires each pair of participants to

```
Function PreComputation()
{
        Phase 0
}

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

Function IP(d_0)
{
 d ← d_0
 CP(d)                 # Calculates β and R_a^2 for
                       the model with attributes d.
 R_af^2 ← R_a^2
 β_f ← β               # R_af^2 and β_f represent the
                       final model outcomes
For every v ∈ D − d_0 do
        {
         CP(d ∪ v)
         If | R_af^2 − R_a^2 | is significant then
                {
```

Fig 1. Algorithm flowchart

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.

In [9], the inversion is done using an iterative method requiring two secure multiparty matrix multiplications for up to 128 iterations when using their settings for Paillier. This makes their protocol quite demanding on the data owners.

In [8], the authors present a generalization to k parties of the secure matrix sum inverse protocol of [12]. This allows them to compute the inverse in one step, which is an improvement on the inversion of [9], but their matrix inversion still requires around k^2 secure 2-party matrix multiplications.

The third protocol was presented in [13], 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.

5. PRELIMINARY

In what follows we propose a privacy preserving regression protocol. We first introduce the properties of the public key cryptosystems that are used in our protocol.

Given a message m_i, we denote the ciphertext by $c_i = Enc_{pk}(m_i)$ where pk is the public key used in the encryption. We will simply use $Enc(m_i)$ when pk is clear from the context. In our protocol, we use Paillier cryptosystem [14] for the case where $l = 1$ and threshold Paillier cryptosystem [15] when $l > 1$ for their 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 $Enc_{pk}(m_i + m_j) = Enc_{pk}(m_i) \times Enc_{pk}(m_j)$ [14]. Moreover, Paillier allows a limited form of homomorphic multiplication, in that we can multiply an encrypted message by a plaintext. It is done as follow: $Enc(m_i)^{m_j} = Enc(m_i m_j)$. 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 [15], [16]. 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 [15] 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 [17]. 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 cyphertext by their secret key. The product of these shares is then computed individually 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 [15].

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 computations are done once at the beginning of the Protocol.

The CP protocol is executed several times for different subsets $d \subseteq D$, it computes β and $R^2{}_a$ for the given d with the collaboration of l out of k data warehouses, say $DW_1,...,DW_l$. Phase 1 of CP is dedicated to the calculation of the regression coefficients β and Phase 2 is dedicated for the calculation of $R^2{}_a$.

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 [12], [17].

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 Basic Functions

The protocol uses several basic functions:

1. Creating Random Matrices, 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 $B_1B_2...B_l$.

2. 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 $b_1b_2...b_l$.

3. Encryption and Decryption Functions for matrices, or $Enc(M)$ and $Dec(M)$, respectively encrypts and decrypts the entries of a matrix M. This is an extension of the regular encryption and decryption functions on integers. Note that l data warehouses will be involved in the decryption.

4. Right Matrix Multiplication Sequence Function, or *RMMS*($B, Enc(M)$), computes $Enc(MB)$. It is done as follow. 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.

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

6. Integer Multiplication Sequence Function, or *IMS*($b, Enc(m)$), 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,…, DW_l, and the result $Enc(mb)$ is sent back to the evaluator.

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:

1. **Computations of $Enc(X^TX)$ and $Enc(X^TY)$:** 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 $Enc(X^TX) = Enc(\sum_{i=1}^{k} {X_i}^T X_i)$, and $Enc(X^TY) = Enc(\sum_{i=1}^{k} {X_i}^T Y_i)$.

2. **Computation of $\sum_{i=1}^{n}(y_i - \bar{y})^2$:**

   1. Each data warehouse DW_i sends their encrypted local aggregate $\frac{\phi_i}{n} = \sum_{i=n_{i-1}+1}^{n_i} \frac{y_i}{n}$ to the Evaluator. The Evaluator homomorphically adds these values to get $Enc(\bar{y}) = Enc(\sum_{i=1}^{n} \frac{y_i}{n})$, and then calculates $Enc(a\bar{y}) = Enc(\bar{y})^a$.

   2. The Evaluator initiates *IMS*($b, Enc(a\bar{y})$) and receives $Enc(ba\bar{y})$. He then initiates $Dec(Enc(ba\bar{y}))$ to get $ba\bar{y}$.

   3. The Evaluator computes $Enc(b^2a^2\bar{y}^2)$ and initiates $\mathrm{IMS}(\frac{1}{b^2}, Enc(b^2a^2\bar{y}^2))$ which results in $Enc(a^2\bar{y}^2)$. The Evaluator then computes $Enc(\bar{y}^2)$.

   4. The Evaluator propagates $Enc(a\bar{y})$ to all data warehouses

5. Each data warehouse computes $Enc(a\delta_i) = Enc(\sum_{i=n_{i-1}+1}^{n_i}(-2ay_i\bar{y}))$

$= \prod_{i=n_{i-1}+1}^{n_i} Enc(a\bar{y})^{-2\sum y_i}$

and $Enc(\alpha_j) = Enc(\sum_{i=n_{i-1}+1}^{n_i} y_i^2)$. Both are sent back to the Evaluator.

6. The Evaluator computes $Enc(a\delta) = Enc(\sum_{i=1}^{k} a\delta_i)$ and recovers $Enc(\delta)$. The Evaluator then proceeds to compute

$Enc(\alpha) = Enc((\sum_{j=1}^{k}\alpha_j) - \delta + n\overline{y^2})$

$= Enc(\sum_{i=1}^{n}(y_i - \bar{y})^2)$

6.3 Protocol: $CP(d)$

First, given d, the evaluator extracts the encryptions of $Z' = X^{dT}Y$ and $Z = X^{dT}X^d$ from $Enc(X^TX)$ and $Enc(X^TY)$ respectively. Then the Evaluator initiates *CRM*(d) and *CRI*.

6.4 Phase 1: Computing β

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

1. The Evaluator computes $Enc(ZA)$, initiates *RMMS*($B, Enc(ZA)$) and receives $Enc(ZAB)$.
2. The evaluator initiates $Dec(Enc(ZAB))$ and receives ZAB.
3. 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 $Enc(Z')$.
4. The Evaluator obtains $Enc(A^{-1}\beta)$ by initiating *LMMS*($B, Enc(B^{-1}A^{-1}\beta)$) and computes $Enc(\beta)$ homomorphically.
5. The Evaluator initiates $Dec(Enc(\beta))$, recovers β and sends it to all data warehouses.

6.5 Phase 2: Computing Ra2

This is dedicated for the calculation of adjusted R^2 measure given by equation (3):

1. As each data warehouse DW_i knows β, they calculate their local residuals: $\zeta_i = \sum_{i=n_{i-1}+1}^{n_i}(y_i - \hat{y}_i)$, encrypt it and send it to the Evaluator. The Evaluator then adds the local residuals homomorphically to obtain $Enc(\zeta) = Enc(\zeta_1) \times \ldots \times Enc(\zeta_k)$.
2. The Evaluator computes $Enc(r\zeta)$ and $Enc(a\alpha)$. He then initiates *IMS*($b, Enc(r\zeta)$) and *IMS*($b, Enc(a\alpha)$) and receives $Enc(br\zeta)$ and $Enc(ba\alpha)$ respectively.
3. The Evaluator then initiates a decryption round for $Enc(br\zeta)$ to obtain $br\zeta$, and uses it to compute $ba\zeta = br\zeta\frac{a}{r}$.
4. The Evaluator now calculates $Enc(R_a^2)$ homomorphically as follows: $Enc(R_a^2) = Enc(1) - Enc(ba\alpha)^{\frac{(n-1)}{(n-d-1)ba\zeta}}$
5. The Evaluator initiates $Dec(R_a{}^2)$ and propagates the result to the different warehouses.

6.6 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 Phase 0, step 2.2 can be replaced by "The Evaluator sends $Enc(a\bar{y})$ to DW_1, who decrypts to obtain $a\bar{y}$. DW_1 then compute and send $ba\bar{y}$ 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 follow the protocol correctly and should not communicate secretly outside the protocol.

6.7 Protocol Modification

In order to perform linear regression, 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 R_a^2 at each iteration of the CP protocol (refer to Step 2.1). With some changes to the protocol, it is possible for the data warehouses to send their full 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 perform Step 1 of Phase 2 without the involvement of the $k-l$ data warehouses. In other words, the Evaluator can calculate $Enc(\zeta_i)$ using $\beta, Enc(X_i),$ and $Enc(Y_i)$ as follows:

$$Enc(\hat{y}_i) = \prod_{j=1}^{d} Enc(x_{ij}{}^d)^{\beta_j} = Enc(\beta x_i{}^d)$$

and

$$Enc(\zeta_i) = Enc(\prod_{=n_{i-1}+1}^{n_i} Enc(y_i) \times Enc(-\hat{y}_i))$$

$$= Enc(\sum_{i=n_{i-1}+1}^{n_i} (y_i - \hat{y}_i))$$

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 encrypted n records have to be stored and then used to calculate $Enc(\zeta_i)$.

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 known.

Since Paillier is semantically secure [14] and since we need l parties to complete decryption, no information can be gained from the ciphertexts exchanged throughout the protocol. We note that when $l=1$, all but one of the ciphertexts sent to DW_1 for obfuscation will be decrypted at the next step, so DW_1 can only gain additional knowledge from a cyphertext if he decrypts $Enc(a\alpha)$ in Phase 2.2. As such, we will look at the decrypted values each party obtains.

In Phase 0, all the values are encrypted except for $ba\bar{y}$, that the l active data owners and the evaluator receive. If the corrupted data owners and the evaluator collaborate, they can remove a and at most $l-1$ b_i's. They can obtain $b'\bar{y}$, but since b' is a random unknown integer, no information about $\bar{y}$ can be recovered. In the case where $l=1$, the active data owner obtains $a\bar{y}$ from the decryption, but since a is random, no information is gained. The same reasoning is true for Phase 2, where all the information is encrypted except for $br\zeta$, which is always obfuscated by at least one random integer. In the case where $l=1$, DW_1 can also obtain $a\alpha$ by performing an extra decryption, but no information about α or ζ 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 party and the evaluator obtain in step 1.2. The evaluator and the corrupted party can obtain ZAB', where $B' = B_1 B_2 ... B_i$ and DW_i is an incorruptible party. Since B' is random, the evaluator cannot recover B_i or Z, even while knowing $AB_1B_2...B_{i-1}$.

We thus have that all the values in the protocol are either encrypted, obfuscated by some random element or sent to all the parties. As such, our scheme is secure and private, since no party can learn additional information from the protocol other than the final result of the computations.

8. COMPLEXITY

In this section, we evaluate the computational complexity and the amount of messages sent during one iteration of our protocol. We will evaluate the individual burden of each participating party as well as the total complexity. Let d be the number of attributes used for a given iteration and let D be the total number of attributes considered.

Our result will be given using some basic functions as the units. More specifically, we will give how many encryptions, decryptions, homomorphic multiplications (HM) and homomorphic additions (HA).

If we are using an instance of Paillier with modulus m^2, we have that HA is equivalent to multiplying two integers modulo m^2, while HM is equivalent to computing an exponentiation modulo m^2. It follows that an encryption is equivalent to 2HM and 1HA, while a standard decryption is essentially 1HM [14].

Finally, in our setting, a (k,l)-threshold decryption is equivalent to having each of the l involved party compute one HM and l HA as well as send $l-1$ messages. It follows that, since $l << m$, HM dominates the decryption and the l HA have a negligible impact on the complexity [18]. As such, we can reasonably assume that (k,l)-threshold decryption is bounded

above by a constant computational complexity of 2HM, making it only slightly more expensive than standard decryption.

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

1. $Enc(M)$ and $Dec(M)$: These functions involve d^2 encryption and decryptions, respectively. If the result is sent, it also requires a total of d^2 and $(l^2+1)d^2$ messages, respectively.
2. RMMS($B, Enc(M)$) and LMMS($B, Enc(M)$): In these functions, each party sends d^2 messages to exactly one other party, for a total of $(l+1)d^2$ messages. The data owner DW_i has to homomorphically compute $Enc(MB'B_i)$, with each entry of this matrix requires at most d HM and $(d-1)$ HA. Thus, the whole matrix requires at most d^3 HM and $d^2(d-1)$ HA for each DW_i.
3. IMS($b, Enc(m)$): Each party sends one message to exactly one other party, for a total of $l+1$ messages. Each data owner DW_i has to homomorphically compute $Enc(mb'b_i)$, which requires one *HM*.

We now evaluate the complexity of each phase, starting with Phase 0. Recall that this phase is all about pre-computations. As such, it will not affect the complexity of an iteration of Phase 1 and 2.

1. Each data owner first computes Z_i and Z'_i. They are then required to send D^2+D+2 encryptions (steps 0.1, 0.2.1 and 0.2.5) and compute 1 HM in phase 0.2.5. Each of the l active data owners also participate in 2IMS and 1 decryption.
2. The evaluator has to perform a total of $(k-1)(D^2+D+3)+2$ HA, 3HM and 1 encryption. He sends a total of $2k+3$ messages.

We now evaluate the complexity of the main protocol, starting with Phase 1. We will not take the generation of the random integers and random matrices into account, since they can be generated and stored ahead of time. In Phase 1, the passive data owners do not participate. Each of the l active data owners participate in 2MMS and $2d^2$ decryptions.

The evaluator performs d^2 encryptions, inverts one plaintext matrix, and sends messages in 2MMS and $2d^2$ decryptions. He also computes a total of (d^3+2d^2) *HM* and $(d^2+2d)(d-1)$ *HA* in steps 1, 3 and 4.

Phase 1 involves a total of $2d^2$ decryptions, $(3d^3+2d^2)$ HM, $(3d^2+2d)(d-1)$ HA and one matrix inversion. A total of $(l^2+2l+3)d^2+(l^2+1+k)d$ messages are sent among all the parties.

In Phase 2, all data owners compute $X\beta$ and ζ_i, and perform one encryption. Each of the l active data owners also participate in 2IMS and 2 decryptions. The Evaluator computes a total of k HA and 3HM. The Evaluator also performs a plaintext multiplication in step 2.3. A total of $2(k+l+l^2+1)$ messages are sent among all the parties.

As such, assuming a decryption is at most 2HM, the final complexity of $CP(d)$ for each participating party is bounded above by the following.

1. All data owners: 2 matrix multiplications, 1 encryption. Sends 1 message.
2. Active data owners additionally have: $2(d^3+d^2+1)+(2d^2+2)$ HM, $2d^2(d-1)$ HA. Sends $(l+1)(d^2+d+1)+1$ messages.
3. The Evaluator: 1 matrix inverse, 1 plaintext multiplication, (d^3+2d^2+3) HM, $(3d^2+2d)(d-1)+k$ HA. Sends $(2d^2+2d+kd+4)$ messages.

As can be seen from this evaluation, if we fix the dimension d, the total complexity of the scheme is linear in k, while the total number of messages is $O(l^2+k)$. The Evaluator absorbs most of the computational complexity, leaving the data warehouses with a complexity depending only on the size d of the matrices, if we assume $l << m$. 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).

Finally, we will compare the complexity of our scheme to that of the schemes of [9] and [8] for each individual participants. For this we shall look mostly at the secure multiparty matrix multiplication protocol of [12]. In the 2-party case, one party has to compute about $3d^2$ HM and d^2 HA for encryption and decryption while the second party has to execute about d^3 HM

and d^3 HA for the homomorphic matrix multiplication and share splitting. As such, in the k-party protocol we can expect an average of $\frac{k}{2}(d^3 + 3d^2)$ HM, $\frac{k}{2}(d^3 + d^2)$ HA and kd^2 messages 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 but the Evaluator.

9. CONCLUSIONS AND FUTURE WORK

We presented a practical system that performs linear regression for a large number of data warehouses without learning anything about the data apart from the regression parameters and diagnostics.

Different from existing 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.

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 we will be using is the IBM Cloud at Western University.

10. 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. Informatics 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] D. C. Montgomery, E. A. Peck, and G. G. Vining, Introduction to linear regression analysis, vol. 821. Wiley, 2012.

[11] G. A. Seber and A. J. Lee, Linear regression analysis, vol. 936. John Wiley & Sons, 2012.

[12] 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.

[13] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-Preserving Ridge Regression on Hundreds of Millions of Records," 2012.

[14] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Advances in cryptology—EUROCRYPT'99, 1999, pp. 223–238.

[15] 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.

[16] Y. Desmedt, "Threshold cryptosystems," in Advances in Cryptology—AUSCRYPT'92, 1993, pp. 1–14.

[17] T. Nishide and K. Sakurai, "Distributed paillier cryptosystem without trusted dealer," in Information Security Applications, Springer, 2011, pp. 44–60.

[18] H. Cohen, A Course in Computational Algebraic Number Theory. Springer-Verlag, 1993.