Privacy-Preserving Logistic Regression

Saeed Samet
eHealth Research Unit, Faculty of Medicine, Memorial University, St. John's, NL, Canada
Email: ssamet@mun.ca

***Abstract*—Logistic regression is an important statistical analysis methods widely used in research fields, including health, business and government. On the other hand preserving data privacy is a crucial aspect in every information system. Many privacy-preserving protocols have been proposed for different statistical techniques, with various data distributions, owners and users. In this paper, we propose a new method to securely compute logistic regression of data, privately shared among two or more data owners. Using this secure protocol, data users can receive the coefficient vector of logistic regression from the data owners, who jointly execute a privacy-preserving protocol, in which only encrypted values are exchanged between them. At the end of the protocol, each data owner will send her portion of the final results to the user to construct the final query result. We have tested our method along with the secure building blocks using sample data to illustrate the performance of the results in terms of computational and *communication* complexities.**

***Index Terms*—privacy-preserving, secure multiparty computation, cryptography, homomorphic encryption, statistical analysis**

I. Introduction

As more data are collected, there is increasing number of demands to extract various knowledge by researchers, such as in health services research and public health systems. Remote access to information over the Internet is one of the current solutions to apply data privacy. However, this method will cause security risks and it requires patient consent. In addition, remote access cannot be used when data are stored and kept by multiple parties. Another privacy-preserving technique is data de-identification, in which different generalization and suppression will be applied on individual records before any data disclosure. Secure multi-party computation, using various cryptosystems is another privacy-preserving method. Using this approach, data users are able to receive aggregate knowledge, while no private and sensitive individual data will revealed to them.

By using secure computation instead of de-identification methods, there is no re-identification risk, and the final results will be in high precision, which is not the case when generalization and suppression are being used. The only disadvantage of secure computation approach is that it is slower compare to data de-identification, because of the time needed for data encryption and decryption. In this paper we use secure computation techniques to compute an important statistical analysis method, Logistic Regression. In our protocol two types of parties are involved; Data Users or Researchers, and Data Owners or Custodians. Each data owner has a subset of records from the whole dataset, such that no other party has access to that portion of plain data. Data user sends her information request to the data owners and receives the final results as some private shares from them. Another advantage of our proposed method is that in large data custodians, such as government statistics organizations, which provide access to potentially identifiable data, data users have to take oath of confidentiality and background checks. It also needs physical presence of the data users at the data centers to perform data analysis. Using our approach allows the data users to perform data analysis remotely. Also, some custodians disclose de-identified version of original data that reduces data utility and reliability. In statistical analysis methods, we are dealing with different computations with various mathematical operations such as addition, multiplications, exponentiation and natural logarithms. To have a secure computation for this set of operations we need secure and efficient privacy homomorphism techniques to apply on those methods. However, most of the homomorphic encryption methods only support one operation, addition or multiplication. The existing fully homomorphic cryptosystems have various security vulnerabilities and are not suitable for privacy-preserving protocols.

Following is the outline of this paper: Section 2 discusses background and related work. Section 3 is dedicated to secure building blocks that are used in the main protocol. Privacy-preserving logistic regression protocol is proposed in Section 4. Section 5 shows the security and complexity analysis of the protocol. Experimental results will be illustrated in Section 6, followed by the conclusions and future work in Section 7.

II. Background and Related Work

Two papers [1], [2], with the title "Privacy-Preserving Data Mining", were published in 2000, in which two different approaches, secure multiparty computation and randomization, have been proposed. After those papers, research in the field of secure computation has rapidly grown and many privacy-preserving methods and protocols have been introduced for statistical analysis and data mining methods [3], [4].

Manuscript received May 29, 2015; revised July 25, 2015.

doi: 10.12720/jait.6.3.88-95

Many secure building blocks, with various security and efficiency levels have been proposed, such as secure sum [5], secure comparison [6], and secure multi-party multiplication and factorial [7]. Most SMC protocols assume that the parties involved are in equal situations, in that they are both data providers and data consumers. These protocols are therefore not applicable in a situation where data custodian only provides the data, and data recipient (say, a researcher) does not provide or have access to data but only wishes to perform statistical analysis methods and/or receives analytical information.

In the existing secure protocols with client-server approach, every time a query is submitted, data owner will use the original data to perform the analysis requested. This may increase the risk of data leakages and privacy breaches. In our method instead the original data will be put off the network after encryption and all the analysis methods will be done on the encrypted information. This will increase the security confidence of the protocols and will justify our solution in many applications, which need high level of privacy-preserving of the original individual data, like health information records. To follow this approach, a homomorphic encryption is needed to support both addition and multiplication operations. Therefore, popular homomorphic cryptosystems like Paillier, RSA, and Elgamal could not be used. On the other hand, the only practical and fully homomorphic encryption system, Ferrer, suffers from some existing cryptanalysis and could not be utilized in secure protocols, as it is presented. To prevent those types of attacks we apply this cryptosystem in a distributed configuration among two or more parties. By using this configuration, no single party has informed about the original data and its encryption value, and all the computations are done jointly on the encrypted information by all the parties involved in the protocol.

Following the Yao's protocol for secure computations [6], which is called the Millionaires problem, using garbled circuit, this method has been used as the base technique in some papers. The most recent work using that technique is done by Ben-david *et al.* [8], called FairplayMP, which is a generalization of an earlier work, Fairplay - a secure two-party computation system [9]. The main issues of this technique are the number of gates required for each operation, and number of data inputs provided by the parties involved. Regarding the number of gates for the final circuit, although it is claimed that the circuit depth has insignificant effect on the overall runtime of the protocol, the number of gates even for a very simple computation like an auction among 10 bidders will result a compiled circuit with 1380 gates. Thus for a more complex computation, such as the statistical analysis methods mentioned in this paper, we will face with circuits of very large number of gates. Another issue, regarding the number of input data, is that the proposed method will not be efficiently applicable when each party has a large number of inputs. This is because every single input owned by each party has to be shared among all the other parties, bit by bit. This needs a huge number of initial communications between the parties, which is not efficient and applicable for real-world applications with large datasets.

Comparing our work to all the existing methods on secure multi-party computation techniques and privacy-preserving statistical analysis methods, the main advantages are de-centralization of original data, keeping the original data off the network, and performing all the computations, to extract the query result, among the data owners in a distributed way using homomorphic encryption and secure building blocks to preserve data privacy.

III. Privacy-Preserving Building Blocks and Preliminaries

Inside the main protocol we need to perform some secure mathematical operation, such as addition, dot product, as well as secure comparison. In this section, we first explain the cryptosystem we use inside the protocol. Then, we show the Secure Addition (SA) building block that we later use inside the main protocol. Secure Comparison will be explained inside the main protocol, and readers can refer to [10] for the detail of Secure Dot Product (SDP).

The encryption technique used in this privacy-preserving protocol is homomorphic encryption. Among the existing homomorphic cryptosystems, such as Paillier [11], RSA [12], and Elgamal [13] we use Paillier, which is an additive homomorphic encryption, i.e. it maps addition of plaintexts to the multiplication of their corresponding ciphertexts. We also utilize Secure Multiparty Addition [7], to add the feature of multiplicative homomorphism in our proposed method. As it mentioned above, in Paillier cryptosystem for any two plaintext messages m_1 and m_2, and their encryptions, $Enc(m_1)$ and $Enc(m_2)$, (1) is maintained:

$$Dec(Enc(m_1) \times Enc(m_2)) = m_1 + m_2 \qquad (1)$$

in which Enc and Dec indicate encryption and decryption, respectively. This cryptosystem has also the following characteristic:

$$Dec(Enc(m_1)^{m_2}) = m_1 \times m_2 \qquad (2)$$

A. Secure Multiparty Addition (SMA)

By using SMA, *n* parties $P_1, P_2, \dots, P_n$ having their own private input shares, $x_1, x_2, \dots, x_n$, will securely compute their private output shares, $y_1, y_2, \dots, y_n$, such that:

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} y_i \qquad (3)$$

This sub-protocol preserves the privacy of the inputs and outputs of the data owners. Mathematical operations in this protocol are modular. For instance, if the operations are done in mod $n = 77$, then $27 + 32 = 34 \times 4$ because $34 \times 4 = 136 = 59 \bmod 77$. In real-world applications, however the number selected for mod is huge, to make sure that the protocol is secure. In our protocols, *n* is the multiplication of two primes, with the length of 1024 in binary system.

B. Dealing with Real Numbers

Note that in the public key cryptosystems all operations are done on integer numbers. Therefore, to overcome this restriction when we are dealing with real numbers in the applications, we should scale the numbers before the encryption. Then, the final results should be rescaled back to reach the correct values after the decryption. Following are two examples:

Suppose two numbers, $a = 1.483$ and $b = 52.37$, are the original data. In the setup phase $a' = scale(a, d) = 1483$, $b' = scale(b, d) = 52370$, and the scaling factor $d = 3$. Then a' and b' will be encrypted, and $Enc(a')$ and $Enc(b')$ will be stored. Now, in the operation phase we want to compute their mean value. Following would be the major steps:

$Enc(1483) \times Enc(52370) = Enc(53853)$ is computed and sent back for decryption. After the decryption, the final result will be multiplied by 10^{-d} and divided by 2 to reach the mean value.

$$\frac{Dec(Enc(53853)) \times 10^{-3}}{2} = \frac{53.853}{2} = 26.9265 = \frac{a+b}{2}$$

Now, suppose we want to compute $\frac{a}{b}$. Following would be the major steps:

- $Enc(1483)$ and $Enc(52370)$ will be decrypted
- $b^{-1} = 1.9094 \times 10^{-5}$ will be computed
- $b' = scale(round(b^{-1})) = 19095, d = 9$
- $\frac{a}{b} = a' \times b' = 28317885$ will be computed
- $E(28317885)$ and $d = 9$ are returned
- After the decryption, the final result will be multiplied by 10^{-d}:

$$Dec(Enc(28317885)) \times 10^{-9} = 0.028317885 \approx \frac{a}{b}$$

C. Preliminaries

First we discuss about the protocol's configuration in terms of the parties involved and their communications. In the main scenario we have three types of parties involved, Central Data Custodian (CDC), Data User (DU), and two or more Data Providers (DP). DPs are working in between the two other parties, by receiving the private shares of data from the CDC and performing the distributed computations to extract and send the query results requested by the DU.

Following notations are used in the rest of the paper:

a: Raw data stored by CDC

Enc_i : Encryption using the public key initiated by DP_i

Dec_i : Decryption using the private key initiated by DP_i

The main characteristics of the protocol are as follows:

- The protocol ensures that the DU cannot get access to or view any individual raw data.
- CDC gives service to the DU through two or more DPs.
- DPs have no access to the original data.

We assume that there is no collusion among all the DPs, and otherwise the original data could be compromised. However, even if all except one DP are illegally collaborated, they won't be able to reach the original data owned by the CDC. Security and collusion attack analysis will be discussed more in detail at the end of this section. There are two main steps in the protocol:

1) Setup phase (Secure data distribution)

In the setup phase CDC creates and transfers the private shares to the DPs and sends metadata and data map to the DU. Fig. 1 illustrates the setup phase of the protocol.

Secure data distribution: In this step each data item a will be privately distributed among N data providers, $DP_1, \dots, DP_N$, as follows:

- Each DP generates a set of public and private keys, and broadcasts her public key.
- CDC randomly selects numbers $a_1, \dots, a_{N-1}$, and calculates a_N such that:

$$a = \sum_{i=1}^{N} a_i \quad (4)$$

- CDC encrypts each private share using the corresponding public key received from the DPs, and sends $Enc_i(a_i)$ to DP_i, for $i = 1, \dots, N$.
- Each DP decrypts the received value from the CDC and privately stores in her own dataset.
- CDC will also send the metadata and data map to the DU, which will be later used to make queries by the DU.

2) Operation phase (Secure shared analysis)

In the second phase, queries requested by the DU will be performed on the secure data by the DPs and the final result will be received by the DU. Fig. 2 illustrates the operation phase of the protocol.

- DU will create her query and will send it to the DPs interfaces.
- According to the query, the DPs will perform the query using Paillier cryptosystem and secure building blocks.
- At the end of the computations, the DU will receive the final result wrapped with her private random value from DPs, and will extract the query result by unwrapping the received value.

IV. Privacy-Preserving Logistic Regression

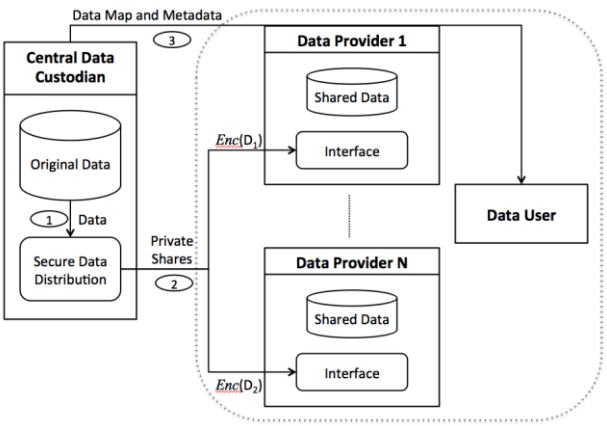

Figure 1. Setup Phase (Secure Data Distribution)

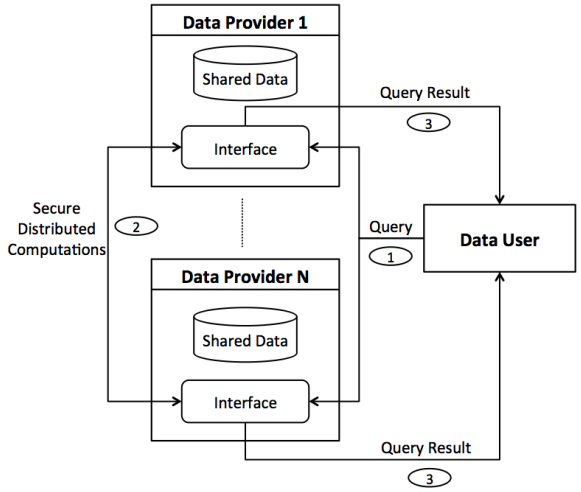

Figure 2. Operation Phase (Secure Shared Analysis)

Without loss of generality in the following algorithms we assume there are two DPs, DP_1 and DP_2, and they could be simply generalized to apply for more than two DPs. Note that to store the categorical data, we first create a map such that each possible type is shown by a number, and then data is privately shared among the DPs. To model a categorical dependent variable using two or more independent variables, logistic regression is used instead of linear regression when the variables have discrete categorical values, and it is actually a generalization of linear models.

In this model, maximum likelihood estimation is utilized to estimate the parameters that best fit the data.

Suppose there is a dataset with M independent samples, and a random variable Z with two possible values, Yes and No. The number of distinct possible combinations is denoted by N, and n_i is the number of samples with the combination i (population i). Also, y is a column vector with i elements representing the number of records with Yes value for Z, $\pi_i$shows the probability of Zbeing Yes for the i -th population, i.e. $\pi_i = P(Z_i = 1|i)$, and Kdenotes the number of independent variables. We also show the design matrix of independent variables by X, which is composed of N rows and $K + 1$ columns, with the value 1 for the first element in each row. We have to compute a vector β with $K + 1$ elements, one for each independent variable, plus one intercept. By using the Newton-Raphson method and the steps in [14] we will reach to computation of β using (5).

$$\beta_z = \beta_{z-1} + [X^T W X]^{-1} X^T (y - \mu) \qquad (5)$$

In (5) $\mu_i = n_i \pi_i$, β_0 is the initial approximation for β, and W is a diagonal $N \times N$ matrix with the elements $n_i \pi_i (1 - \pi_i)$ on its diagonal and zero elsewhere.

This computation will be iterated until there is no change between the corresponding elements in β_z and β_{z-1}, by considering a specific threshold.

$$W = \begin{bmatrix} n_1\pi_1(1-\pi_1) & 0 & \cdots & 0 \\ 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & n_N\pi_N(1-\pi_N) \end{bmatrix}_{N\times N}$$

$$X = \begin{bmatrix} 1 & a_{1,1} & \cdots & a_{1,K} \\ 1 & a_{2,1} & \cdots & a_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & a_{N,1} & \cdots & a_{N,K} \end{bmatrix}_{N\times(K+1)}, \quad \mu = \begin{bmatrix} n_1\pi_1 \\ \vdots \\ n_N\pi_N \end{bmatrix}_{N\times 1}$$

$$\mu_i = \frac{e^{X_i \times \beta_z}}{1+e^{X_i \times \beta_z}}, \quad w_{i,i} = \frac{e^{X_i \times \beta_z}}{(1+e^{X_i \times \beta_z})^2} \qquad (6)$$

Before continuing the protocol, we show the secure computation of matrix multiplication when each matrix is privately shared between two parties. Suppose there are two matrices, $P_{s,t}$ and $Q_{t,u}$, as follows:

$$P = \begin{bmatrix} p_{1,1,1} + p_{1,1,2} & p_{1,2,1} + p_{1,2,2} & \cdots & p_{1,t,1} + p_{1,t,2} \\ p_{2,1,1} + p_{2,1,2} & p_{2,2,1} + p_{2,2,2} & \cdots & p_{2,t,1} + p_{2,t,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s,1,1} + p_{s,1,2} & p_{s,2,1} + p_{s,2,2} & \cdots & p_{s,t,1} + p_{s,t,2} \end{bmatrix}_{s\times t},$$

$$Q = \begin{bmatrix} q_{1,1,1} + q_{1,1,2} & q_{1,2,1} + q_{1,2,2} & \cdots & q_{1,u,1} + q_{1,u,2} \\ q_{2,1,1} + q_{2,1,2} & q_{2,2,1} + q_{2,2,2} & \cdots & q_{2,u,1} + q_{2,u,2} \\ \vdots & \vdots & \ddots & \vdots \\ q_{t,1,1} + q_{t,1,2} & q_{t,2,1} + q_{t,2,2} & \cdots & q_{t,u,1} + q_{t,u,2} \end{bmatrix}_{t\times u}$$

In which $p_{i,j,k}$, $q_{i,j,k} \in DP_k$. Now, we compute $R_{s,u} = P \times Q$ such that:

$$R = \begin{bmatrix} r_{1,1,1} + r_{1,1,2} & r_{1,2,1} + r_{1,2,2} & \cdots & r_{1,u,1} + r_{1,u,2} \\ r_{2,1,1} + r_{2,1,2} & r_{2,2,1} + r_{2,2,2} & \cdots & r_{2,u,1} + r_{2,u,2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{s,1,1} + r_{s,1,2} & r_{s,2,1} + r_{s,2,2} & \cdots & r_{s,u,1} + r_{s,u,2} \end{bmatrix}_{s\times u}$$

In matrix R, $r_{i,j,k} \in DP_k$, such that:

$$r_{i,j} = r_{i,j,1} + r_{i,j,2} = \sum_{m=1}^{t} (p_{i,m,1} + p_{i,m,2}) \times (q_{m,j,1} + q_{m,j,2})$$

$$= \sum_{m=1}^{t} (p_{i,m,1} \times q_{m,j,1} + p_{i,m,1} \times q_{m,j,2} + p_{i,m,2} \times q_{m,j,1} + p_{i,m,2} \times q_{m,j,2}) \qquad (7)$$

The first and last terms of (7) will be locally computed by each corresponding DP. SDP is utilized to securely compute the second and third terms as follows:

$$\sum_{m=1}^{t} (p_{i,m,1} \times q_{m,j,2}) = c_{i,j,1} + c_{i,j,2} \qquad (8)$$

$$\sum_{m=1}^{t} (p_{i,m,2} \times q_{m,j,1}) = d_{i,j,1} + d_{i,j,2} \qquad (9)$$

such that $c_{i,j,k}, d_{i,j,k} \in DP_k$. Therefore, matrix R will be converted to the summation of two matrices, each of which belongs to one DP:

$$R = R_1 + R_2$$

$$= \begin{bmatrix} r_{1,1,1} & \cdots & r_{1,u,1} \\ \vdots & \ddots & \vdots \\ r_{s,1,1} & \cdots & r_{s,u,1} \end{bmatrix} + \begin{bmatrix} r_{1,1,2} & \cdots & r_{1,u,2} \\ \vdots & \ddots & \vdots \\ r_{s,1,2} & \cdots & r_{s,u,2} \end{bmatrix} \quad (10)$$

Inside the algorithm of computing the logistic regression, we also need to find the inverse of a matrix in the form of the summation of two private matrices each of which belongs to one DP, same as the above matrix R. For this building block we utilize the privacy-preserving protocol proposed by Han et al. in [15], by which the two DPs are jointly compute their private shares as follows:

$$[R_1 + R_2]^{-1} = V_1 + V_2 \quad (11)$$

Now, back to the main protocol, each β_z has to be jointly computed by the DPs, in a way that no party revealed her private data to the others. Therefore, we have to use a privacy-preserving computation to apply on (11). For instance π_i, the probability of Zbeing Yes for the i-th population is computed from the samples, which their information is distributed among the DPs. Thus, $\mu = n_i \pi_i$ and W are not owned by a single party. In the first matrix operation of (5), $[X^T W X]^{-1}$, X is distributed among the DPs. Therefore, by using the above secure building block $X^T W X$ could be jointly computed by the DPs and will be converted to two separate private shares as follows:

$$[X^T W X]^{-1} = [R_1 + R_2]^{-1},\ R_i \in DP_i \quad (12)$$

Note that to compute $X_i \beta_z$, SDP could be used. Next step is to compute $[X^T W X]^{-1}$, which is now converted to $[R_1 + R_2]^{-1}$. For this part, as mentioned above, proposed protocol in [15] will be used, such that at the end each DP_i has her own private matrix, V_i, such that:

$$[R_1 + R_2]^{-1} = V_1 + V_2\ ,\ V_i \in DP_i \quad (13)$$

For the second part of (5), i.e. $X^T (y - \mu)$, secure matrix multiplication is used for $X^T y$ and $X^T \mu$, and consequently $X^T (y - \mu)$ will be converted to the addition of two separate and private matrices owned by the DPs, such that:

$$X^T (y - \mu) = W_1 + W_2,\ W_i \in DP_i \quad (14)$$

Now, we have the following matrix operation:

$$(V_1 + V_2) + (W_1 + W_2) = V_1 W_1 + V_1 W_2 + V_2 W_1 + V_2 W_2) \quad (15)$$

By using privacy-preserving matrix multiplication for $V_1 W_2$ and $V_2 W_1$ in (15), the whole equation will be converted to the summation of two separate matrices, each one privately owned by one DP as follows:

$$[X^T W X]^{-1} X^T (y - \mu) = Z_1 + Z_2,\ Z_i \in DP_i \quad (16)$$

Therefore, β_z is jointly adjusted at the end of each iteration, by the DPs:

$$\beta_z = \beta_{z-1} + Z_1 + Z_2 \quad (17)$$

in which β_{z-1} is also the summation of two separate matrices privately owned by the DPs. Now, by comparing the corresponding items in β_z and β_{z-1}, DPs can figure out the termination of the iteration. To compare the two vectors, we can use the Euclidean distance (2-norm distance). However, because each vector is privately shared between the two DPs, a privacy-preserving method has to be applied. The two DPs will perform the following algorithm to compute the 2-norm distance between two vectors, say $\alpha = \alpha_1 + \alpha_2$and $\beta = \beta_1 + \beta_2$:

$$\|\alpha + \beta\| = \sqrt{\sum_{i=1}^{K} \left((\alpha_{i,1} + \alpha_{i,2}) - (\beta_{i,1} + \beta_{i,2})\right)^2}$$

$$= \sqrt{\sum_{i=1}^{K} \left((\alpha_{i,1} - \beta_{i,1}) + (\alpha_{i,2} - \beta_{i,2})\right)^2}$$

$$\sum_{i=1}^{K} \left((\alpha_{i,1} - \beta_{i,1}) + (\alpha_{i,2} - \beta_{i,2})\right)^2 = \sum_{i=1}^{K} (\delta_{i,1} + \delta_{i,2})^2$$

$$= \sum_{i=1}^{K} \delta_{i,1}^2 + \sum_{i=1}^{K} \delta_{i,2}^2 + 2 \sum_{i=1}^{K} \delta_{i,1} \delta_{i,2} \quad (18)$$

when $(\alpha_{i,j} - \beta_{i,j} = \delta_{i,j})$.

According to (18), DPs have to run a SDP to get their own private shares for $2\sum_{i=1}^{K} \delta_{i,1} \delta_{i,2}$ as follows:

$$2 \sum_{i=1}^{K} \delta_{i,1} \delta_{i,2} = b_1 + b_2 \quad (19)$$

Each DP_j , sets her private share as

$$c_j = b_j + \sum_{i=1}^{K} \delta_{i,j}^2 \quad (20)$$

Now, we have to compare the computed distance, $c_1 + c_2$, with some constant h as the threshold value for the termination of the loop. Following are the steps for this secure comparison:

Secure Comparison:

- DP_1 encrypts her private value, c_1, and sends $Enc(c_1)$ to DP_2.
- DP_2 generates a private random number, r_2 , encrypts her private value and the constant value, k , and sends $(Enc(c_1) \times Enc(c_2) \times Enc(k)^{-1})^{r_2}$ to DP_1.
- DP_1 decrypts the received value and compares it with zero. If the decrypted value is less than zero, it means that the 2-norm distance between the two vectors is less than k.

After computing the final shares of the regression coefficients vector β, now it has to be securely sent to the DU. Therefore, DPs will perform following steps:

- DU generates a vector of random numbers:

$$R_{DU} = \langle r_1, \cdots, r_K \rangle \quad (21)$$

And sends their encryptions to DP_2.

- DP_2 encrypts items of her private vector β_2 and sends the following list to DP_1:

$Enc(\beta_2) \times Enc(R_{DU})$

$$= \langle Enc(\beta_{1,2}) \times Enc(r_1), \cdots, Enc(\beta_{K,2}) \times Enc(r_K) \rangle \quad (22)$$

- DP_1 decrypts each item of the received list, adds the corresponding item from her vector β_1 to it and sends the following list to the DU:

$$\langle \beta_{1,1} + \beta_{1,2} + r_1, \cdots, \beta_{K,1} + \beta_{K,2} + r_K \rangle \quad (23)$$

- DU subtracts each of the random numbers of the vector R_{DU} from the corresponding item in the received list to find the final vector for the regression coefficient, β.

V. Security and Complexity Analysis

A. Security Analysis

Our assumption is that all the trusted third parties involved are semi-honest, i.e. each party properly follows all the steps of the protocol, communicates and exchanges correct data with other parties, while she may utilize the intermediate information to reach others' private data. Simulation paradigm [16,17] along with the composition theorem are used for the security proof of the protocol. By using the simulation paradigm, a protocol is known secure if all the received data by a party can also be obtained by her input and output. Thus, for each party P we have to find a simulator S such that its output is computationally indistinguishable [16] from that party's view using the secure protocol. Composition theorem is also utilized in the security proof of the protocols because they are usually complex and each protocol is composed of some sub-protocols such that the input of one sub-protocol is the output of the previous one. Here, we analyze the security of the setup phase, which is the fundamental part of the protocol, secure distribution of the original data to the DPs by the CDC. Then we investigate the security of the algorithms in the operation phase. We also show the security analysis of the secure two-party multiplication that we have used as a secure building block in some algorithms. Security proof of the SDP could be found in its corresponding paper [10]. Finally, we discuss collusion attacks between the parties.

Setup phase: In this step, each data value is securely distributed by CDC among the DPs. Therefore, we have to create one simulator for each party involved in this step.

Data distribution: CDC in this phase has the original data a as her private input, and will create private shares $a_1, \ldots, a_N$ for the DPs. Thus, the simulator S_0 for this party in the protocol would be:

Input: $\{a\}$

Process: Selecting random numbers $a_1, \ldots, a_{N-1}$

Computing a_N such that $a = \sum_{i=1}^{N} a_i$

Output: $a_1, \ldots, a_N$

$DP_1, \ldots, DP_N$: During the setup phase, Each DP_i will only receive a share of the data. Therefore, the simulator S_i for this party is:

Input: A random number $\{a_i\}$

Process: Nothing

Output: Nothing

Secure Two-party Addition: Suppose, there are two parties, P_1 and P_2, each of which has a private input, x_1 and x_2 respectively. We denote the protocol of secure two-party addition by STA and the desired functionality of STA by $f(x_1, x_2)$, such that:

$$f: X \times X \rightarrow Y \times Y \quad (24)$$

Furthermore, we show the first and second elements of $f(x_1, x_2)$ by $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, which are the private outputs of P_1 and P_2 respectively. Also, the view of P_1 (respectively P_2) during the execution of STA on (x_1, x_2) are denoted by $VIEW_1^{STA}(x_1, x_2)$ (respectively $VIEW_2^{STA}(x_1, x_2)$). Therefore, we have (25) and (26):

$$VIEW_1^{STA}(x_1, x_2) = \{x_1, y_1, E_1(y_1, e_1)\} \quad (25)$$

$$VIEW_2^{STA}(x_1, x_2) = \{x_2, y_2, E_1(x_1, e_1)\} \quad (26)$$

Note, that as the P_2's point of view, $E_1(y_1, e_1)$ is just a random number generated by and received from P_1 to that party. We say that protocol STA privately computes F, if two polynomial-time algorithms V_1 and V_2 exist, such that:

$$\{V_1(x_1, f_1(x_1, x_2))\} \triangleq \{VIEW_1^{STA}(x_1, x_2)\} \quad (27)$$

$$\{V_2(x_2, f_2(x_1, x_2))\} \triangleq \{VIEW_2^{STA}(x_1, x_2)\} \quad (28)$$

In (28), symbol $\triangleq$ means that the two ensembles in both sides of the equivalence symbol are computationally indistinguishable.

Proof: For V_1, suppose P_1 is corrupted by an adversary. Thus, the adversary knows all the items in the set of P_1's view in the protocol STA, i.e. the set of $\{x_1, y_1, E_1(y_1, e_1)\}$. Obviously, simulator V_1 would be trivially as follows:

Input: (x_1, y_1)

Process: Computing $E_1(y_1, e_1)$

Output: $\{x_1, y_1, E_1(y_1, e_1)\}$

For the simulator V_2, suppose P_2 is corrupted by an adversary. P_2's view in STA is the set of $\{x_2, y_2, E_1(x_1, e_1)\}$ in which, as it is previously indicated, $E_1(x_1, e_1)$ is considered a random number received by P_2. Therefore, the simulator V_2 has to generate a random number and send it to P_2 each time P_2 needs to get a message from P_1. Thus, V_2 could be as follows:

Input: (x_2, y_2)

Process: Generating a random number r

Output: $\{x_2, y_2, r\}$

As it is shown above we can conclude that the incoming messages of P_1 and P_2 in the protocol STA and incoming messages of P_1 and P_2 from V_1 and V_2, respectively, are indistinguishable.

A very similar proof could be shown for multi-party case and other protocols in this paper, by utilizing composition theorem [10], [18], which is used in the protocols that contain secure building blocks as sub-protocols.

Collusion attacks: One security problem in all distributed privacy-preserving protocols is the risk of colluding two or more parties involved in the protocols against the other parties. In our protocol this might happen between two or more DPs to reach the original data, which is securely kept by CDC. However, collusion of all the trusted third parties is needed to compromise the original information kept by the CDC.

B. Complexity Analysis

To analyze the complexity of the algorithm presented in the main protocol for logistic regression, we separate different operations performed by the parties involved. The main operations are:

1. Messages sent from one party. A message means one block of values sent from one party to another party
2. Values sent from one party, i.e. each single value sent by a party to another party
3. Encryption
4. Decryption
5. Local mathematical operations, such as addition, multiplication, etc.

The first two operations are considered as the communication costs and the rest of the operations are related to the computational costs. We investigate the complexity of the algorithm proposed in the main protocol, which helps to understand its performance for using in the any dataset. In the following tables we assume that there are n records in the dataset, r is the selected power, and N and K are defined in the logistic regression algorithm. Table I shows the complexity analysis for the secure building blocks. Using the complexity of the secure building blocks, the main algorithm of the protocol could be analyzed in terms of complexity. Table II illustrates communication costs for the algorithm, in terms of the number of messages exchanged between the parties involved. Number of encryptions and decryptions, and also local computations are shown on the Tables III and IV, respectively. Note, that most of the local computations are negligible comparing with other computational costs and communication costs and could be performed offline and in parallel. Although the encryptions and decryptions are also performed locally, we separate them from the rest of the local computations because of the time needed for these types of operations, and some of them have to be sequentially done in certain steps, such as the decryptions. In Table V, the number of each distributed operation is shown for the algorithms. $\mathrm{SDP(n)}$ denotes secure dot product of two n-dimensional vectors.

TABLE I. COMPLEXITY ANALYSES OF THE SECURE BUILDING BLOCKS.

Algorithm	Sent Messages	Sent Values	Enc	Dec	Local Operations
SA	2	2	2	1	*: 1, ^: 1
SDP	2	n+1	n+1	1	*: n, ^: n

TABLE II. COMMUNICATION COSTS OF THE ALGORITHM.

Algorithm	Sent Messages	Sent Values
Logistic Regression	$4KN + 4K^2 + 16K + 16$	$2KN^2 + 2K^2 + 6KN + 6K^2 + 16K + 8$

TABLE III. COMPUTATIONAL COSTS OF THE ALGORITHMS (ENCRYPTIONS AND DECRYPTIONS).

Algorithm	Encryptions	Decryptions
Logistic Regression	$8K^2N + 5KN + 3K^2 + 2K + 3$	$5KN + 3K^2 + K + 1$

TABLE IV. COMPUTATIONAL COSTS OF THE ALGORITHMS (LOCAL COMPUTATIONS).

Algorithm	Local Computations
Logistic Regression	$+, -: 2(\mathrm{n} + \mathrm{K} + 1)$, $*: 8\mathrm{K}^2 + \mathrm{K} + 2$, $\wedge: 8\mathrm{K}^2 + 2\mathrm{n} + 1$

TABLE V. DISTRIBUTED OPERATIONS BETWEEN THE PARTIES.

Algorithm	Number of distributed operations
Logistic Regression	$5KN \times \mathrm{SDP(K)} + 3K^2 \times \mathrm{SDP(N)}$

VI. EXPERIMENTAL RESULTS

To measure the performance of the methods we have implemented the protocol and its secure building blocks using Java programming language. To investigate the performance of the proposed protocols, we first implement the secure building blocks. To have a high performance when dealing with very large numbers we use BigInteger class, which provides efficient arithmetic operations for very large numbers. We have used Mac OS X, Intel® Core™ i5 2.60GHz, 8 GB DDR3 RAM for the experiment. Table VI shows the performance of the encryption, decryption and secure building blocks. The encryption key length is 1024 bits. Table VII illustrates the overall time for the protocol. The number of records is 100,000 and the number of variables is 10. To have a better performance we can optimize the distributed operations among the parties, especially between the two DPs, in one or more of the following ways:

1. Initial encryptions in each DP and random number generations can be done in advance. This approach is used in the secure building blocks, such as SDP.
2. Previously computed values could be stored and used for the future requests.
3. Block data transferring between the DPs.

VII. CONCLUSION AND FUTURE WORK

We have proposed secure method for the computation of logistic regression that is used in many applications, especially in health records, which need to maintain the privacy of the patient's information. During the execution of the protocol, the original cleartext of the data will be kept off the network, and the computation is done by two

or more semi-trusted third parties on the private distributed data, and the final results will only be decrypted and sent back to the end-user. As a future work, we are currently extending our research to cover other important statistical methods, which are extensively used in health research. Also, collaboration of the multiple data custodians could be done homogenously or heterogeneously. We are also working on other homomorphic cryptosystems to use in our secure protocols for a better performance in terms of speed and coverage of various mathematical operations.

TABLE VI. PERFORMANCE RESULTS FOR SECURE BUILDING BLOCKS.

Algorithm	Time (in Seconds)
Encryption or Decryption	0.00157
SA	0.00314
SDP	13.677

TABLE VII. PERFORMANCE RESULTS FOR THE PROTOCOLS.

Algorithm	Overall Time (in Seconds)
Logistic Regression	3524

ACKNOWLEDGMENT

This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Research & Development Corporation of Newfoundland and Labrador (RDC).

REFERENCES

[1] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Proc. the 20th Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, USA, 2000, pp. 36-54.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. the ACM Special Interest Group on Management of Data Conference (SIGMOD)*, Dallas, TX, USA, 2000, pp. 439-450.

[3] J. Vaidya, C. W. Clifton, and M. Zhu, *Privacy Preserving Data Mining*, Springer, 2006, pp. 120.

[4] C. C. Aggarwal and P. S. Yu, *Privacy-Preserving Data Mining: Models and Algorithms*, Springer, 2008, pp. 513.

[5] C. Clifton, *et al.*, "Tools for privacy-preserving distributed data mining," *SIGKDD Explorations*, Newsletter, vol. 4, no. 2, pp. 28-34, 2002.

[6] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annual Symposium on Foundations of Computer Science (SFCS)* IEEE Computer Society: Washington, DC, USA, 1982, pp. 160-164.

[7] S. Samet and A. Miri, "Privacy-preserving bayesian network for horizontally partitioned data," in *Proc. 2009 IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2009)*, Vancouver, Canada, 2009, pp. 9-16.

[8] A. Ben-david, N. Nisan, and B. Pinkas, "FairplayMP: A system for secure multi-party computation," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2008. Alexandria, VA, USA: ACM.

[9] D. Malkhi, *et al.*, "Fairplay—a secure two-party computation system," in *Proc. 13th Conference on USENIX Security Symposium*, Berkeley, CA, USA: USENIX Association, 2004.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, Prague, Czech Republic, 1999, pp. 223-238.

[11] A. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, 1978, pp. 120-126.

[12] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, pp. 469-472, 1985.

[13] S. A. Czepiel. (2002). Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation. [Online]. Available: http://czep.net/stat/mlelr.pdf

[14] S. Han, W. K. Ng, and P. S. Yu, "Privacy-preserving linear fisher discriminant analysis," in *Proc. 12th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, Springer-Verlag: Osaka, Japan, 2008, pp. 136-147.

[15] O. Goldreich, "Foundations of cryptography," *Basic Applications*, vol. 2, New York, NY, USA: Cambridge University Press, 2004.

[16] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186-208, 1989.

[17] B. Goethals, *et al.*, "On private scalar product computation for privacy-preserving data mining," in *Proc. 7th International Conference on Information Security and Cryptology (ICISC)*, Seoul, Korea, vol. 3506, 2004, pp. 104-120.

[18] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143-202, 2000.

Saeed Samet is a member of the Faculty of Medicine as an Assistant Professor at the e-Health Research Unit. He has designed and developed several secure protocols for various health applications. He received his Ph.D. in Computer Science from the University of Ottawa in 2010 and his thesis was "Privacy-Preserving Data Mining", in which he has proposed and designed different protocols on privacy-preserving data mining and machine learning methods and techniques.