Shi *et al. BMC Medical Informatics and Decision Making* 2016, **16**(Suppl 3):89
DOI 10.1186/s12911-016-0316-1

BMC Medical Informatics and Decision Making

RESEARCH Open Access

Secure Multi-pArty Computation Grid LOgistic REgression (SMAC-GLORE)

Haoyi Shi[1,2], Chao Jiang[1,3], Wenrui Dai[1], Xiaoqian Jiang[1], Yuzhe Tang[2], Lucila Ohno-Machado[1] and Shuang Wang[1*]

From The 5th Translational Bioinformatics Conference (TBC 2015)
Tokyo, Japan. 7-9 November 2015

Abstract

Background: In biomedical research, data sharing and information exchange are very important for improving quality of care, accelerating discovery, and promoting the meaningful secondary use of clinical data. A big concern in biomedical data sharing is the protection of patient privacy because inappropriate information leakage can put patient privacy at risk.

Methods: In this study, we deployed a grid logistic regression framework based on Secure Multi-party Computation (SMAC-GLORE). Unlike our previous work in GLORE, SMAC-GLORE protects not only patient-level data, but also all the intermediary information exchanged during the model-learning phase.

Results: The experimental results demonstrate the feasibility of secure distributed logistic regression across multiple institutions without sharing patient-level data.

Conclusions: In this study, we developed a circuit-based SMAC-GLORE framework. The proposed framework provides a practical solution for secure distributed logistic regression model learning.

Background

Biomedical research can benefit from data sharing from distributed sources. For example, comparative effectiveness research requires data comparison among different data sources to determine which existing health care interventions work best for certain patients. This requires a large amount of data to be harmonized. Big biomedical data networks, such as the patient-centered SCAlable National Network for Effectiveness Research (pSCANNER) clinical data research network (CDRN) [1], the Scalable Architecture for Federated Translational Inquiries Network (SAFTINet) [2] and the Electronic Medical Records and Genomics (eMERGE) Network [3] have been established to enable cross-institutional biomedical studies. However, information exchange of biomedical data (e.g., genome sequences, diagnoses, medication information, etc.) can put patient privacy at risk, where the potential risks include, but are not limited to, denial of certain types of insurance [4]. As a result, research participants may lose trust in research institutions, which may have an adverse impact on biomedical research. Privacy risks in biomedical studies have been demonstrated in many recent studies. For example, Vaidya demonstrated the possibility to re-identifying individuals from the public query system of the Healthcare Cost and Utilization Project (HCUP) [5]. Sweeney's study successfully identified participants of the Personal Genome Project (PGP) [6]. In addition, with some background information, an attacker can even identify sensitive information of a participant using "anonymized" biomedical data [7–9]. The study in [10] demonstrated that aggregated genome statistics (e.g., allele frequencies) can present privacy risks. Therefore, it is imperative to develop privacy-preserving techniques to facilitate biomedical research. For this purpose, many distributed model learning frameworks [11–18] have been proposed for building a global model involving multiple participants, but without sharing sensitive patient-level information.

In this paper, we consider the scenario of horizontally partitioned data, where different institutions possess data from different patients, but with the same variables. Our previous work, Grid binary LOgistic REgression model (GLORE [11]) was developed to allow sharing models without necessarily sharing patient data in a distributed manner. It leveraged the

* Correspondence: shw070@ucsd.edu
[1]Department of Biomedical Informatics, University of California, San Diego, CA 92093, USA
Full list of author information is available at the end of the article

aggregation of non-sensitive decomposable intermediary results to build a shared model. Its Bayesian extension, EXPLORER [19], proposed online learning to update the model with incremental data. It enabled asynchronous communication to alleviate probable service breakdown when coordinating multiple participants. Recently, grid ordinal and multinomial logistic regressions [13] were developed to deal with multi-center modeling with multiple categorical values for response variables. Remarkably, distributed modeling learning can also be employed in Cox regression for survival analysis [20–22].

Existing privacy-preserving solutions to multi-site regression [23, 24] can guarantee the precision of model learning. However, patient information could leak in these solutions due to disclosure of the information matrix and score vectors during iterative model learning [25, 26]. To protect these exchanged data, many secure multi-party computation (SMC) methods [18, 27–33] have been developed for distributed model learning. Unfortunately, existing SMC-based methods would still suffer from inappropriate disclosure under certain conditions due to the secure sum protocol. Therefore, El-Eman et al. [15] proposed the SPARK protocol that utilized different secure blocks to build a secure distributed logistic regression, which aims to offer stronger privacy protection for patient data. Although, homomorphic encryption based systems [34–36] can protect secure outsourcing, they need to assign the same public keys in the case of multi-party computation, which may leak intermediary results during communications among participants.

In this paper, we propose a secret-sharing circuit-based secure multi-party computation framework for grid logistic regression (SMAC-GLORE). Inheriting the distributed model learning framework from GLORE, SMAC-GLORE protects not only patient-level data, but also all the intermediary information exchanged during the model learning phase. Introducing secure multi-party computation to build boolean circuits for private data in learning, the proposed framework prevents participants from interpreting arbitrary intermediary information, such as aggregation of summary statistics, and only releases the final learned model parameters.

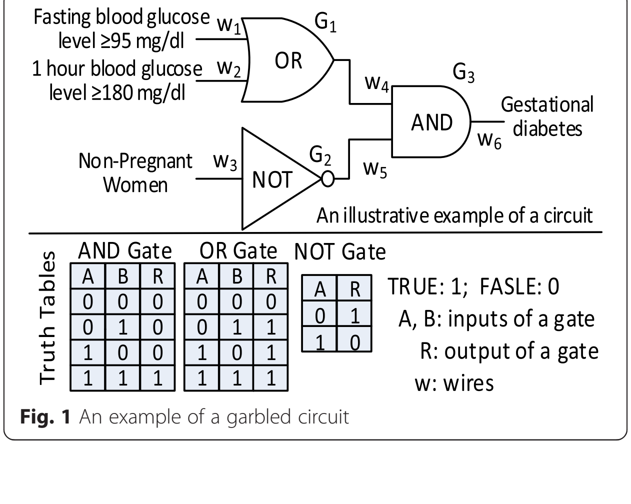
Methods

To securely evaluate the logistic function, we introduced secret-sharing circuits-based Secure Multi-party Computation (SMC) into procedure of calculation. SMC provides a method for parties to jointly compute a function over their data while keeping the data private in semi-honest scenarios, where all the participants are those who are honest in running programs and algorithms correctly, but might be curious about the information transferred among entities.

Garbled circuits

The key idea of circuit based computation is based on the fact that operations in almost all modern digital computers are implemented by circuits combining basic logic gates such as AND, OR, NOT etc., where inputs and outputs of a gate may be TRUE or FALSE for certain propositions. One can design a garbled circuit counterpart [37–40] to protect the data and the computation. Figure 1 shows an example of diagnosing gestational diabetes based on blood glucose level (BGL) in a standard circuit representation, which consists of three gates (i.e., G_i with $i = 1,2,3$) and six wires (i.e., w_j with $j = 1, 2,\ldots, 6$). Using Boolean algebra [41] and truth tables of the three basic gates shown in the figure, the circuit can calculate (Gestational diabetes) = (NOT (Non-Pregnant Women)) AND ((Fasting BGL ≥95 mg/dl) OR (1 h BGL ≥180 mg/dl)). In theory, one can build circuits of any complexity using basic logical gates to evaluate functions (e.g., secure distributed logarithm, exponent, etc.) or algorithms (e.g., secure distributed logistic regression models in this paper).

A garbled circuit [37, 39] is a specially designed circuit, which enables two (or more) parties to securely compute a function $f\,(x_A,\ x_B)$ without exposing their private secrets (e.g., x_A and x_B are inputs from party A and party B, respectively). All parties here are assumed to be semi-honest collaborators, which means that they follow the protocol honestly, but may try to deduce additional information from the received messages during the protocols' execution. Figure 2 illustrates several key steps to securely compare two integers using a garbled comparison circuit (GCC), where the comparison function can be formulated as $f(x_A,\ x_B) = \begin{cases} 1, & if\ x_A \neq x_B \\ 0, & otherwise \end{cases}$ with $x_A, x_B \in \{0, 1, 2, 3\}$. The first step is to convert the integer inputs into a binary representation, where we use 2 bits to encode 4 different integers, i.e., {0 = 00; 1 = 01; 2 = 10, 3 = 11}. In the second step, a computation service provider (CSP) needs to build a circuit to implement the comparison function. The circuit consists of two XOR gates (i.e., G1 and G2) and performs bitwise comparisons between x_A and x_B, where a XOR gate outputs 1 if



AND Gate

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate

A	R
0	1
1	0

Fig. 1 An example of a garbled circuit

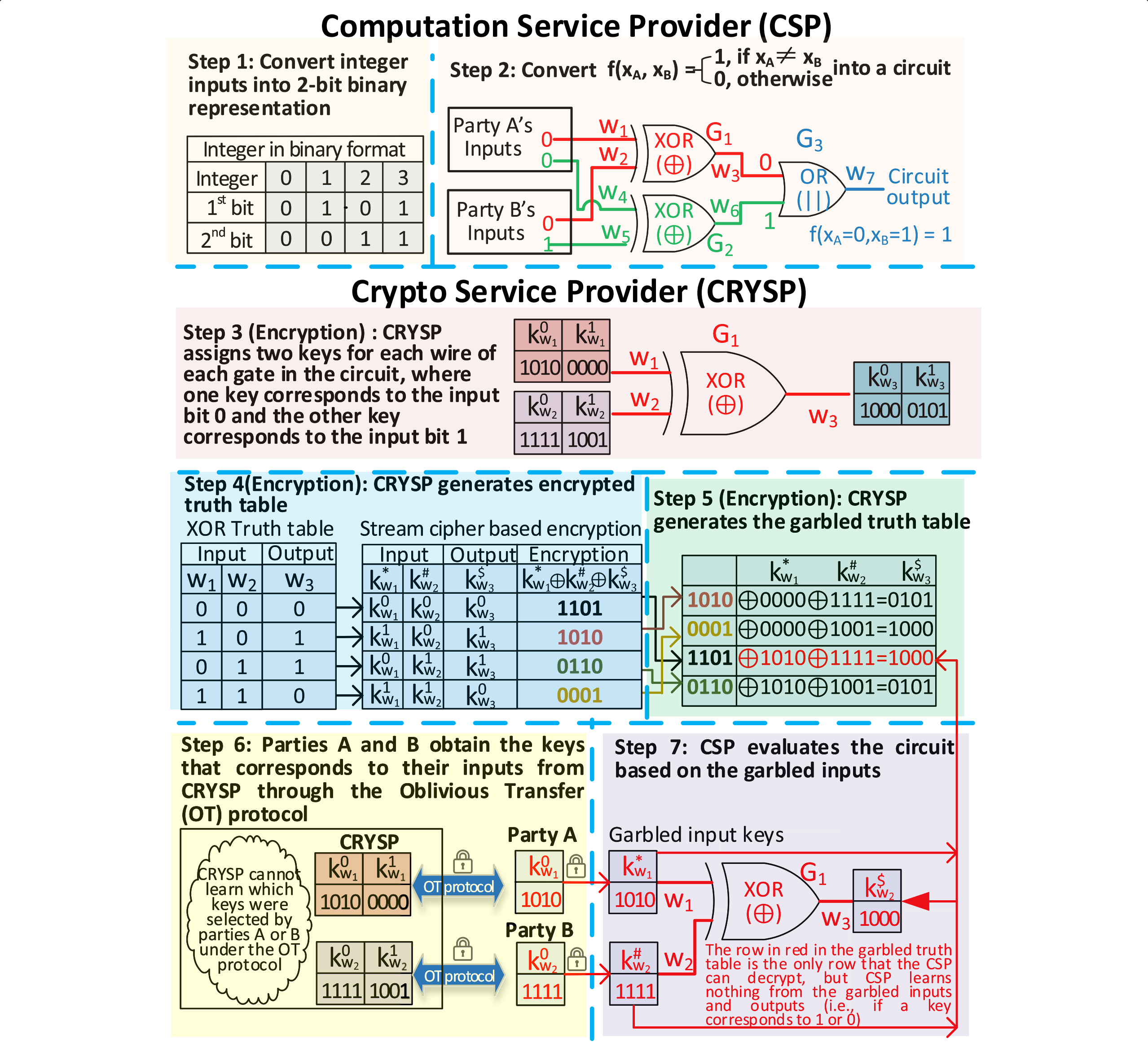

Computation Service Provider (CSP)
Step 1: Convert integer inputs into 2-bit binary representation
Integer in binary format
Integer 0 1 2 3
1st bit 0 1 0 1
2nd bit 0 0 1 1
Step 2: Convert f(xA, xB) = 1, if xA ≠ xB; 0, otherwise into a circuit
Party A's Inputs
Party B's Inputs
W1 W2 W3 W4 W5 W6 W7
XOR (⊕)
OR (||)
G1 G2 G3
Circuit output
f(xA=0,xB=1) = 1
Crypto Service Provider (CRYSP)
Step 3 (Encryption) : CRYSP assigns two keys for each wire of each gate in the circuit, where one key corresponds to the input bit 0 and the other key corresponds to the input bit 1
1010 0000
1111 1001
1000 0101
Step 4(Encryption): CRYSP generates encrypted truth table
XOR Truth table
Stream cipher based encryption
Input Output Encryption
1101 1010 0110 0001
Step 5 (Encryption): CRYSP generates the garbled truth table
1010 ⊕0000⊕1111=0101
0001 ⊕0000⊕1001=1000
1101 ⊕1010⊕1111=1000
0110 ⊕1010⊕1001=0101
Step 6: Parties A and B obtain the keys that corresponds to their inputs from CRYSP through the Oblivious Transfer (OT) protocol
CRYSP cannot learn which keys were selected by parties A or B under the OT protocol
CRYSP
OT protocol
Party A
Party B
Step 7: CSP evaluates the circuit based on the garbled inputs
Garbled input keys
The row in red in the garbled truth table is the only row that the CSP can decrypt, but CSP learns nothing from the garbled inputs and outputs (i.e., if a key corresponds to 1 or 0)

Fig. 2 Private computations using garbled circuits, where parties A and B would like to compute whether two integers from them are identical, but without disclosing the actual integer value from each party

its inputs are different, otherwise it outputs 0. The circuit utilizes an OR gate (i.e., G3) to compute whether any output of the two XOR gates is 1. For example, given inputs $x_A = 0$ and $x_B = 1$, the circuit outputs 1 in step 2 of Fig. 2 (i.e., $x_{A \neq} x_B$). In the third step, a crypto service provider (CRYSP) generates two encryption keys for each wire of each gate in the circuit, where one key corresponds to bit 0 and the other to bit 1. Step 3 in Fig. 2 shows an example of assigning six keys for the XOR gate G1 with two input wires and one output wire. For the output wire w_3, $k^0_{w_3}$ and $k^1_{w_3}$ are the randomly assigned keys (e.g., 1000 and 0101 in Fig. 2), which correspond to the wire outputs 0 and 1, respectively. Both keys were randomly chosen by the Crypto Service Provider (CRYSP) during the circuit initialization phase. In step 4, CRYSP will substitute the truth table of each gate with the encryption keys generated in step 3 and encrypt the outputs through a stream cipher based encryption scheme [42]. In step 5, CRYSP will generate a garbled truth table by randomly permuting rows of the encrypted truth table and sending it to the CSP for garbled circuit evaluation. In step 6, parties A and B will securely obtain the corresponding keys of their inputs (e.g., $k^0_{(w_2)} = 1111$ for input bit 0 of wire w_2) from CRYSP through the oblivious transfer (OT) protocol [43], by which CRYSP cannot learn the actual selections of keys from the parties. In step 7, parties A and B will send their garbled keys (e.g., $k^*_{w_1} = 1010$ and $k^*_{w_2} = 1111$) to CSP, but without disclosing whether they refer to bit 1 or bit 0. Therefore, given the two garbled input keys, CSP can only decrypt one row (i.e., $k_{\overline{w_3}}^{=\,1000}$) in the garbled truth table (as shown in red) without learning the underlying true value. Following the same protocol, CSP can evaluate the entire garbled circuit without learning any intermediary information. Finally, CSP sends the garbled key of the final output to CRYSP to find out whether it corresponds to 0 (unmatched) or 1 (matched). For garbled circuit based application, we can consider the circuit initialization phase as the encryption procedure, where encrypted truth table for each gate is created and distributed between two parties through the OT protocol.

In the case in which there is no trusted CRYSP, party A could, for example, serve as the CRYSP and party B could be CSP to avoid potential collusion risk between CSP and CRYSP. Moreover, one can choose more advanced encryption algorithms (other than the streaming cipher used in this example) to achieve a better protected OT protocol. The above example demonstrates how a secure integer comparison function can be achieved using garbled circuit-based SMC. In practice, advanced circuits are required to handle more complicated tasks, such as secure distributed logistic regression, where only the learned model parameters are allowed to be released as circuit outputs.

However, Yao's garbled circuit is only secure in 2-party semi-honest scenarios, which is not sufficient for practical use. Usually, there are more than 2 parties or participants engaged in the same computing task. We based our model on the GMW project developed by Choi [44], who implemented the classical SMC protocols of Goldreich, Micali, and Wigderson (the GMW protocol) [45]. The GMW protocol uses secret-sharing rather than garbled truth tables to implement the secure computation, which enables the computation among more than 2 parties. In GMW, all the variables are represented as binary numbers and the protocol itself is able to protect against a semi-honest adversary with any number of corrupted parties. All the functions should be interpreted as boolean-circuits and each participant feeds the encrypted private data as input to the circuits. During the process of computing, none of the participants can interpret any temporary values except the final output. However, the GMW project only supports non-negative integers. We established our own encoding format, enabling the support for real number arithmetic, and built libraries for secure matrix operation primitives, which made it possible to solve the practical problems of building a secure distributed logistic regression model.

The proposed framework is based on the well-developed GMW protocol [44] for SMC. We overcame the limitations of GMW protocol and built several secure computation primitives to support SMAC-GLORE. As discussed in the original GMW implementation paper [44], the Naor-Pinkas OT was implemented as the encryption scheme to secure the computation. The mathematical definitions and proofs of security of the Naor-Pinkas OT protocol for GMW have been discussed in [46].

Platform preparation

In our project, all the floating values are represented by binary vectors in 28-bits fixed-point format, in which 11 bits are assigned for the fractional part and the other 17 bits for the integer part. The highest bit of the integer part is reserved as a sign of positive or negative value. The two's complement [47] method is adopted to represent negative values. Thus, all possible values under the proposed fixed-point format are ranging from $-(2^{28}-1)\times 2^{-11}$ to $(2^{27}-1)\times 2^{-11}$. The proposed platform extends the integer addition and multiplication to support floating number arithmetic. Here, we describe methods for implementing subtraction and multiplication.

Subtraction

When doing subtractions, we need to iteratively compare bits of minuend and subtrahend. If the bit in minuend is smaller than the corresponding bit in the subtrahend, we may need to borrow a bit leftwards. However, in circuits, it is very difficult and expensive to implement borrowing bits. Therefore, we choose to use two's complement. To calculate the subtraction, we first calculate the two's complement of the subtrahend, where we invert or flip all the bits of subtrahend and then add 1 to the least significant bit. After that, we add the calculated value to the minuend.

Multiplication

In multiplication, the result may, at most, double the number of required bits to represent the product. For example, the product of two n-bits numbers should have at most $2n$ bits. However, the GMW project requires the two factors involved in a multiplication to have the same number of bits, and allocates the same number of bits for the product. So, if we still use 28 bits for multiplication, we will suffer from significant precision loss or computing error. To solve this problem, we double the size of all the values when doing multiplication. In another word, we expand the 28-bits values to 56-bits values before multiplication and use the 56-bits values for the multiplication. As the calculated result will also have 56 bits, we need to drop 28 bits. Even though, in this procedure, we may waste a lot of bits and computing effort, we can secure the computing precision.

Based on the basic operations described above, we built several secure primitives in the GMW project for matrix operations, including matrix addition, matrix subtraction and matrix multiplication, which are elaborated in detail later in this section.

Problem definition

For clarity, in the remainder of this paper, we reserve regular symbols to scalar variables and bold symbols to vectors or matrices. Logistic regression is widely used in biomedical decision support applications, including feature selection, survival analysis, etc. Suppose there is a training dataset $\mathcal{D} = \{(\boldsymbol{X}, \boldsymbol{y})\} = \{(\boldsymbol{x}_1, y_1), \cdots (\boldsymbol{x}_n, y_n)\}$ of n records for patients, where $y_i \in \{0, 1\}$ is the observed binary outcome and $\boldsymbol{x}_i$ corresponds to the m-dimensional covariates for the i-th patient. In the logistic regression model, the likelihood function of $y_i = 1$ given $\boldsymbol{x}_i$ can be expressed as follows,

$$P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}_i}}, \qquad (1)$$

where $\boldsymbol{\beta}$ is a parameter vector that measures the relationship between the response variable y_i and covariates $\boldsymbol{x}i$. Note that $P(y_i = 0|\boldsymbol{x}_i, \boldsymbol{\beta}) = 1 - P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta})$ for binary response variable y_i. Given the training dataset $\mathcal{D}$, $\boldsymbol{\beta}$ can be estimated through maximization of the following log likelihood function

$$\hat{\boldsymbol{\beta}} = \text{argmax}_{\boldsymbol{\beta}} \left(l(\boldsymbol{\beta}) = -\sum\nolimits_{i=1}^{n} \log\left(1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}_i}\right)\right). \qquad (2)$$

Here, we use $l\ (\boldsymbol{\beta})$ to represent the log-likelihood function. Since there is no closed-form solution for $\boldsymbol{\beta}$, iterative numerical solutions are required to obtain the optimal parameters. In a centralized model, the Newton-Raphson method is widely used to find $\hat{\boldsymbol{\beta}}$. The iterative maximization is achieved by calculating the first and second derivatives of the log-likelihood function $l(\boldsymbol{\beta})$. In the t-th iteration, current estimation $\boldsymbol{\beta}^{(t)}$ is updated by

$$\begin{aligned}\boldsymbol{\beta}^{(t+1)} &= \boldsymbol{\beta}^{(t)} - \left[l''\left(\boldsymbol{\beta}^{(t)}\right)\right]^{-1} l'\left(\boldsymbol{\beta}^{(t)}\right) \\ &= \boldsymbol{\beta}^{(t)} + \left(\boldsymbol{X}^T \underbrace{\boldsymbol{W}^{(t)}}_{diag\left(p\left(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)}\right)\left[1 - p\left(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)}\right)\right]\right)} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \left(\boldsymbol{y} - \underbrace{\boldsymbol{\mu}^{(t)}}_{vector\left(p\left(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)}\right)\right)}\right)\end{aligned} \qquad (3)$$

In Eq. (3), $\boldsymbol{W}^{(t)}$ is the diagonal matrix with elements $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)})p(y_i = 0|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)})$ and $\boldsymbol{\mu}^{(t)}$ is the vector of probabilities $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}^{(t)})$. Since the Hessian matrix $\boldsymbol{H} = \boldsymbol{X}^T \boldsymbol{W}^{(t)} \boldsymbol{X}$ is a square matrix of second partial derivatives $l^{''}(\boldsymbol{\beta})$, the iterative procedure can be rewritten as

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \boldsymbol{H}^{-1}\, l^{'}\left(\boldsymbol{\beta}^{(t)}\right). \qquad (4)$$

In this study, we consider a distributed model learning problem, where $\mathcal{D}$ is horizontally partitioned by h parties as $\mathcal{D} = \left\{\left(X^1, , y^1\right), \cdots, \left(X^h, , y^h\right)\right\}$. For the j-th party p_j, $\boldsymbol{X}^j = \left(\boldsymbol{x}_1^j\, \boldsymbol{x}_2^j \cdots \boldsymbol{x}_{n_j}^j\right)$ is a $m \times n_j$ matrix representing the subset of n_j covariates from p_j, and $\boldsymbol{y}^j = \left(y_1^j, y_2^j, \cdots, y_{n_j}^j\right)$ is the vector of n_j corresponding binary response variables. Consequently, the intermediary results for the logistic regression model in Eq. (3) can be linearly decomposed [11] as

$$\begin{aligned}\boldsymbol{X}^T \boldsymbol{W}^{(t)} \boldsymbol{X} &= \sum_{j=1}^{h} \left(\boldsymbol{X}^j\right)^T \left(\boldsymbol{W}^j\right)^{(t)} \boldsymbol{X}^j \\ \boldsymbol{X}^T\left(\boldsymbol{y} - \boldsymbol{\mu}^{(t)}\right) &= \sum_{j=1}^{h} \left(\boldsymbol{X}^j\right)^T \left(\boldsymbol{y}^j - \left(\boldsymbol{\mu}^j\right)^{(t)}\right)\end{aligned} \qquad (5)$$

Equation (5) shows that each party can calculate its own intermediary results conditioned on its local data $(\boldsymbol{X}^j, \boldsymbol{y}^j)$, and share them for the combined results. However, this method requires a trusted server [11] to exchange local statistics. In this paper, we will build a decentralized framework for logistic regression using secret-sharing circuits based on secure multi-party communication. The proposed framework protects the intermediary statistics $(\boldsymbol{X}^j)^T(\boldsymbol{W}^j)^{(t)}\boldsymbol{X}^j$ and $(\boldsymbol{X}^j)^T(\boldsymbol{y}^j - (\boldsymbol{\mu}^j)^{(t)})$ with a joint function for all the parties without disclosing any private information.

Algorithm 1 (A1) summarizes the key steps in the proposed SMAC-GLORE framework. Each participant provides encrypted data $\boldsymbol{X}^j$ as input. The only output of the algorithm is the learned model coefficients $\boldsymbol{\beta}$. All the intermediary information exchange is protected by the OT protocol and secret-sharing circuits. In A1: line 1, each party can locally calculate its own part of the fixed-Hessian matrix $\tilde{H}_i$ (see Eq. (7)) and feed it as part of input to the circuit, while within the circuit, the fixed-Hessian matrix is securely constructed as $\tilde{\boldsymbol{H}} = \sum\nolimits_{i=1}^{h} \tilde{H}_i$ based on Eq. (7). In A1 line 2, we apply the proposed secure matrix inversion algorithm (see Secure Hessian matrix inversion section) to get the inversion of $\tilde{\boldsymbol{H}}$, where we implement the Strassen Algorithm to speed up the matrix multiplication and improve performance. Then we iteratively update $\boldsymbol{\beta}$ until it converges via A1 lines 4–7, where we first securely construct $\boldsymbol{X}$ from each participant's input and repeat the procedure described in Eq. (4). Within each iteration, we update the first derivative of the maximum likelihood function with the current $\boldsymbol{\beta}$ (A1: line 5). In this procedure, reciprocals or divisions are required and we use the same procedure as for matrix inversion.

Algorithm 1. Secure multi-party computation based grid logistic regression

Input: Each party p_j provides encrypted data X^j.

Output: Learned model coefficients β in the proposed SMAC-GLORE framework

```
1. Securely construct approximate Hessian Matrix based on Equation (7).
2. Securely invert the approximate Hessian Matrix based on Equation (8).
3. Initialize $\beta$ as an all-zero vector.
4. Repeat
5.   Securely calculate the first derivative based Equation (13) for all parties
6.   Securely update $\beta$ based on Equation (4).
7. Until parameters converge
```

Secure Hessian matrix inversion

In this study, the distributed Newton-Raphson method [11, 12] is adopted to compute the coefficients β across multiple parties. Since the Hessian matrix H represents the matrix of the second partial derivatives of maximum likelihood function $l(\beta)$, it has to be updated with the most up-to-date $\beta^{(t)}$ and inverted for each iteration. However, calculating the inversion of H would significantly increase the computational burden and could be unfeasible for our secret-sharing, circuit-based method. One workaround to reduce the complexity is to replace the varying Hessian Matrix with a fixed matrix. According to Böhning [48], the true Hessian matrix can be approximated by

$$H = -\frac{1}{4}X^TX \tag{6}$$

where $X = [X^1 \ X^2 \cdots X^h]$. Consequently, the approximated Hessian Matrix can be rewritten as

$$\tilde{H} = -\frac{1}{4}\sum_{j=1}^{h}\left(X^j\right)^T X^j \tag{7}$$

Equation (7) implies that each party p_j can locally calculate its own part of a partial Hessian matrix $\tilde{H}_i = -\left(X^j\right)^T X^j/4$, and feed $\tilde{H}_j$ as part of the input to the circuit. While in the circuit, the approximated Hessian matrix is constructed by aggregating all the partial Hessian matrices as $\tilde{H} = \sum_{j=1}^{h}\tilde{H}_j$.

To further reduce the computational complexity of inverting $\tilde{H}$, we use the approximating method introduced by Nardi [49]. The numerical approximation method iteratively computes $\tilde{H}^{-1}$ according to Eq. (8).

$$\begin{aligned} N_{t+1} &= 2N_t - N_tM_t, & N_0 &= c^{-1}I, \\ M_{t+1} &= 2M_t - M_t^2, & M_0 &= c^{-1}\tilde{H}, \end{aligned} \tag{8}$$

where $M_t = N_t\tilde{H}$, and c is constant. After convergence (e.g., in ~10 to 15 iterations), N_t will provide an accurate approximation to the inversion of $\tilde{H}$.

Matrix multiplication

We transfer the matrix inversion problem into an iterative procedure of matrix multiplication and addition. Therefore, optimizing the implementation of matrix multiplication can improve the efficiency of the proposed framework. In this subsection, we adopted the Strassen algorithm for matrix multiplication.

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix},\ B = \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix},\ C = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}, \tag{9}$$

Let us denote A and B two square $n \times n$ matrices and $C = AB$ their matrix product. Here, n is recommended to be a power of 2, namely $n = 2^k, k \in \mathbb{N}$. A, B and C can be partitioned into equally sized block matrices as where $A_{i,j}$, $B_{i,j}$, and $C_{i,j}$, are all $(n/2) \times (n/2)$ matrices. According to the definition of block matrix multiplication, $C_{i,j}$, can be represented by $A_{i,j}$, and $B_{i,j}$, for $i, j = 1, 2$.

$$\begin{aligned} C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1}, & C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2}, \\ C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1}, & C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2}. \end{aligned} \tag{10}$$

According to Eq. (10), the calculation of C requires the same number of multiplications as the standard definition of matrix multiplication $C = AB$. To reduce the number of multiplications, we introduce the Strassen algorithm which defines some new matrices based on $A_{i,j}$, and $B_{i,j}$.

$$\begin{aligned} &M_1 = A_{1,1}\left(B_{1,2} - B_{2,2}\right), \quad M_2 = \left(A_{1,1} + A_{1,2}\right)B_{2,2}, \\ &M_3 = \left(A_{2,1} + A_{2,2}\right)B_{1,1}, \quad M_4 = A_{2,2}\left(B_{2,1} - B_{1,1}\right), \\ &M_5 = \left(A_{1,1} + A_{2,2}\right)\left(B_{1,1} + B_{2,2}\right), \quad M_6 = \left(A_{1,2} - A_{2,2}\right)\left(B_{2,1} + B_{2,2}\right), \\ &M_7 = \left(A_{1,1} - A_{2,1}\right)\left(B_{1,1} + B_{1,2}\right) \end{aligned} \tag{11}$$

Equation (11) requires only 7 matrix multiplications between $(n/2) \times (n/2)$ square matrices (one for each M_l, $l = 1, \cdots, 7$) to calculate $C = AB$, which reduces the number of multiplications by $n^3/8$. The product C can be recovered from $\boldsymbol{M}_l$, $l = 1, \cdots, 7$ by

$$\begin{aligned} \boldsymbol{C}_{1,1} &= \boldsymbol{M}_5 + \boldsymbol{M}_4 - \boldsymbol{M}_2 + \boldsymbol{M}_6, \quad \boldsymbol{C}_{1,2} = \boldsymbol{M}_1 + \boldsymbol{M}_2, \\ \boldsymbol{C}_{2,1} &= \boldsymbol{M}_3 + \boldsymbol{M}_4, \quad \boldsymbol{C}_{2,2} = \boldsymbol{M}_5 + \boldsymbol{M}_1 - \boldsymbol{M}_3 - \boldsymbol{M}_7. \end{aligned} \tag{12}$$

The matrices can be iteratively partitioned k times, when $n = 2^k$. Thus, the Strassen algorithm can reduce the complexity of matrix multiplication from $O(n^3)$ to $O(n^{2.8})$.

In this work, the Strassen algorithm is implemented for matrix inversion, which has a significant effect on computational complexity. For other ordinary matrix multiplications, we still use the standard method. However, it is also possible to extend the Strassen algorithm to employ it in ordinary matrix multiplication, e.g. multiplication between non-square matrices.

The first derivative of the maximum likelihood function

In the t-th iteration, the first derivative $l'(\boldsymbol{\beta})$ of the maximum likelihood function has to be updated with the current $\boldsymbol{\beta}^{(t)}$. The k-th element of $l'(\boldsymbol{\beta})$ can be obtained in a distributed manner.

$$\frac{\partial l}{\partial \beta(k)}\Big|_{\boldsymbol{\beta}=\boldsymbol{\beta}^{(t)}} = \sum_{j=1}^{h}\sum_{i=1}^{n_i}\left(y_i^j - P\left(\boldsymbol{x}_i^j, \boldsymbol{\beta}^{(t)}\right)\right)x_i^j(k), \tag{13}$$

where $x_i^j(k)$ and $\beta(k)$ are the k-th element of $\boldsymbol{x}_i^j$ and $\boldsymbol{\beta}^{(t)}$, respectively. Equation (13) shows that we can allow each party to separately compute its own part of first derivative based on local data and we add these results [11]. However, such approach will leak the information of $\boldsymbol{\beta}^{(t)}$ at each iteration. Therefore, we need to securely evaluate Eq. (13) without releasing any intermediary $\boldsymbol{\beta}^{(t)}$. As a result, we need to securely evaluate the exponential function e^x in the boolean circuits. In the proposed framework, we use the Taylor series to approximate the evaluation of e^x, such that we only need to handle multiplication and addition operations.

$$e^x = \sum_{i=1}^{\infty}\frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots \tag{14}$$

To simplify computation and avoid overflow, we set a filter to bound the exponential within the interval between −5 and 5. When the exponentials are greater than 5 or smaller than −5, the evaluation results of the logit function (i.e., $1/(1+e^x)$) would be smaller than 6.7×10^{-2} according to Eq. (1). Thus, we will not lose much accuracy by using this bound.

Simulations in MATLAB shows that the Taylor series could achieve an approximated result with an error less than 10^{-2}, when the maximal order for the expansion is set to 15. To reduce the number of multiplications, we transform the Taylor series to a recursive algorithm.

$$e^x \approx 1 + x\left(1 + \frac{x}{2}\left(\ldots\left(1 + \frac{x}{14}\left(1 + \frac{x}{15}\right)\right)\right)\right). \tag{15}$$

We built a look-up table storing the inversion of integers from 1 to 15, to avoid divisions and to speed up the calculation. For the other divisions involved in the logistic function, we treat them as a matrix of size 1.

It is worth mentioning that all the computations in this section are carried out in a customized Boolean circuit, where all the inputs and intermediary information exchange are protected by the OT protocol and the circuits. The only outputs in plaintext are the learned model parameter $\boldsymbol{\beta}$ in the proposed SMAC-GLORE.

Results

In this section, we first describe computational performance evaluations for fundamental operations, including matrix addition, matrix multiplication and matrix inversion. We then describe accuracy evaluations over real datasets with three features, including the Edinburgh dataset, which contains *T wave inversion*, *Sweating* and *Pain in right arm* features, and three genome datasets [50], where the first two features are *ethnicity groups* and the third feature is a SNP. The last column for each dataset is the intercept.

Computational performance evaluation

We first evaluated the performance of matrix addition, matrix multiplication and matrix inversion under a 2-party setup. We varied the size of matrices from 1×1 to 20×20 for each party. We simulated both parties on a 64-bit Ubuntu 14.04 platform with an Intel Xeon CPU at 3.10GHz and 256GB RAM, under the 28-bit fixed-point encoding, where both parties were connected by a 1GB network.

Figure 3 depicts the performance in terms of communication cost (i.e., bar plots) and circuit computation time costs (i.e., line plots) with the same matrix sizes under a 2-party setup for three secure matrix operations. We can see both the computation time and communication costs increase significantly as the matrix sizes increase. This is because the circuit, which implements the functions to be executed, contains more gates as the size of matrix increases. Moreover, secure matrix addition operations have much lower complexity in terms of computation and communication than performing matrix multiplication operations. Among all three secure matrix operations, matrix inversion shows the highest communication and computational demands since it requires several iterations of matrix multiplication and addition operations.

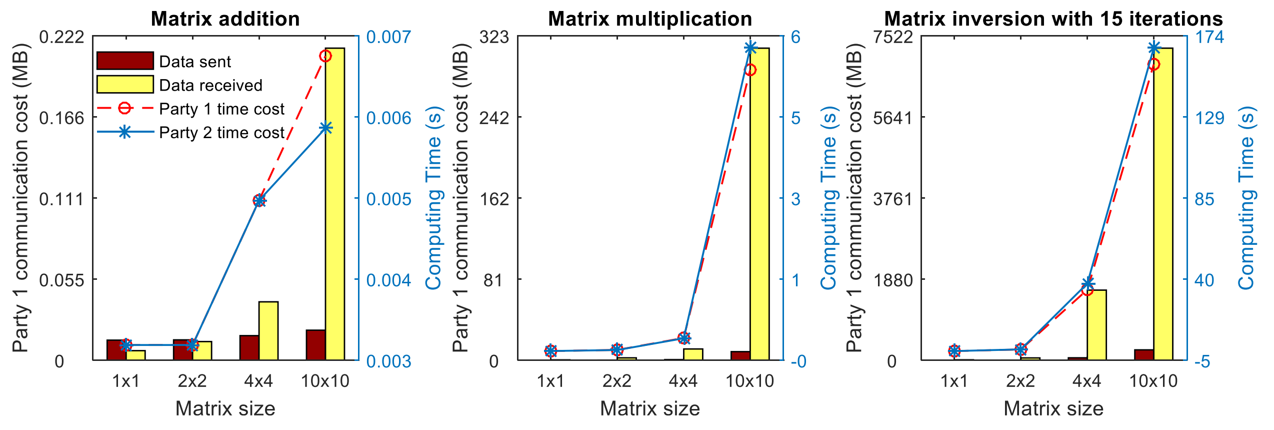


Fig. 3 Computational performance in terms of communication costs (i.e., *bar plots*) and circuit computation time (without OT) (i.e., *line plots*) for matrix addition, matrix multiplication and matrix inversion under a 2-party setup

Table 1 shows the computational performance in terms of number of OT and total time cost (including both OT and circuit evaluation costs) for the operations of matrix addition, matrix multiplication and matrix inversion with different matrix sizes in a 2-party setup. For matrix inversion, we measure all the computational performance with 15 iterations. We also included the size of the circuits for different setups in the terms of the total number of gates and the number of AND gates in Table 1, as the evaluation of AND gates is expensive due to the invocations of OT. We can see that the total time increases linearly with the increase of matrix size for matrix addition operations, while the total time increases exponentially for matrix multiplication and matrix inversion operations. This is because the number of AND gates increases exponentially in matrix multiplication and matrix inversion. In the GMW project, where the XOR gates are "free", only AND gates need secret-sharing evaluations. Therefore, the computing time in matrix multiplication and matrix inversion increases exponentially. Oblivious transfer (OT) is the most computationally expensive part and is the only part which relies on public-key techniques and encryption. During the OT initialization phase, for each wire indexed by w in the circuit, GMW generates a secret share s_{wi} of the input value v_w for each party P_i, $i = 1, \ldots, h$ for a total of h parties, and transfers each share to the corresponding party. During the gate-computation procedure, given a wire w to be evaluated, all the secret shares s_{wi} will be re-collected to securely evaluate the gate. Therefore, we can consider the OT initialization as the encryption time and consider the circuit evaluation part or gate- computation part as the decryption time. We also find that in matrix multiplication and inversion, it takes more time for party 2. The bar plots in Fig. 3 show that party 1 receives much more data than it sends out, which is due to the fact that party 2 serves as the sender in OT while party 1 serves as the receiver.

Accuracy evaluation

In this section, we perform accuracy evaluation of the proposed framework. As we introduced several approximation schemes (e.g., fixed-point encoding format, Taylor expansion for exponential function, division-free matrix inversion, etc.), the accuracy evaluation intends to measure how the results of the proposed framework differ from those of ordinary methods.

Under the same 2-party settings, we first evaluated the accuracy in terms of mean squared error (MSE) between the secure matrix inversion and ordinary matrix inversion operations. The observed MSEs are mainly due to the use of the fixed-point encoding format and the division-free matrix inversion algorithm. Figure 4 depicts the MSE performance with different matrix inversion iterations using 2×2, 4×4, 6×6, 8×8 and 10×10 matrices. We find that as the number of iterations increases, the MSEs decrease for all these matrices, and the MSE can get under 10^{-5}.

We also evaluated the accuracy of learned coefficients $\boldsymbol{\beta}$ in secure distributed logistic regression using different data sets. Our experiments were carried out under two setups: a *local* setting (simulated on a single server, which is the same as the one used in the Computational performance evaluation section) and a *remote* setting (on three different machines). In the *remote* setting, we executed the program on three servers, including the server in the local setting and two other 64-bits Ubuntu servers with Intel Xeon CPUs at 2.40GHz and 96 GB RAM.

In the *local* setting, we simulated both 2-party and 3-party setups. In the *remote* setting with four parties, we hosted the first two parties on a server with 256 GB memory and deployed the other two parties on the remaining two servers. We used the same simulated

Table 1 Computational performance for different matrix sizes in terms of number of gates, OT and total time cost for matrix addition, matrix multiplication and matrix inversion in a 2-party setup

Matrix addition operation						
Matrix size	# of AND gates	# of total gates	OT time (s) Party 1	Party 2	Total time (s) Party 1	Party 2
1 × 1	27	250	0.346	0.194	0.354	0.202
2 × 2	108	994	0.348	0.194	0.357	0.20
4 × 4	432	2,850	0.343	0.202	0.353	0.212
10 × 10	2,700	24,802	0.369	0.230	0.387	0.247
Matrix multiplication operation						
Matrix size	# of AND gates	# of total gates	OT time (s) Party 1	Party 2	Total time (s) Party 1	Party 2
1 × 1	4,621	21,594	0.367	0.245	0.384	0.262
2 × 2	37,076	273,034	0.518	0.609	0.577	0.660
4 × 4	580,325	2,707,002	2.135	3.636	2.603	4.060
10 × 10	4,645,300	21,664,002	21.174	50.413	29.646	58.769
Matrix inversion operation (15 iterations)						
Matrix size	# of AND gates	# of total gates	OT time (s) Party 1	Party 2	Total time (s) Party 1	Party 2
2 × 2	1,030,869	4,872,479	4.864	10.908	6.519	12.472
4 × 4	8,027,793	37,694,207	36.503	85.848	49.619	98.314
6 × 6	26,847,253	125,771,967	121.266	296.780	170.634	349.398
8 × 8	63,345,729	296,412,479	281.653	676.747	405.865	810.214
10 × 10	123,379,701	576,922,463	528.062	1286.500	751.421	1519.897

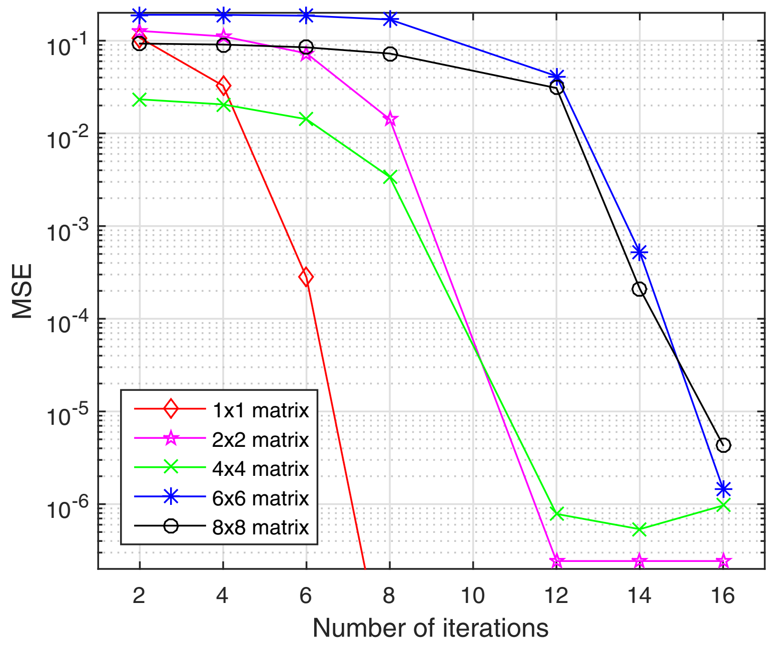


Fig. 4 Number of iterations used in secure matrix inversion and MSE for five different matrices

Table 2 Model parameters β learned in SMAC-GLORE and ordinary logistic regression model

β	2 parties				Ordinary logistic regression				Two-sample Z test	
	Value	Wald test			Value	Wald test			Test statistic	p-value
		SE	Z value	p-value		SE	Z value	p-value		
β_1	−0.6182	0.7759	−0.7968	0.4256	−0.6274	0.7779	−0.8065	0.4199	0.0084	0.9933
β_2	2.5454	0.8461	3.0084	0.0026	2.5767	0.8511	3.0275	0.0025	−0.0261	0.9792
β_3	1.2246	1.1226	1.0909	0.2753	1.2407	1.1369	1.0913	0.2751	−0.0101	0.9920
β_4	0.6177	0.8283	0.7457	0.4558	0.6198	0.8319	0.7450	0.4562	−0.0018	0.9986

dataset for all setups, which contains 60 records and 3 binary features. Table 2 compares the results of the learned coefficients, Wald test and two-sample Z test between the proposed SMAC-GLORE and the ordinary logistic regression model. We can see that the proposed SMAC-GLORE framework achieved similar performances as those of ordinary logistic regression, as the outputs are very close, with coefficient differences between 10^{-3} and 10^{-2}. Table 3 illustrates the results among 2-party, 3-party and 4-party setups with the same datasets. We can see in Table 3 that all these setups generated exactly the same outcomes. The experimental results demonstrated that the proposed SMAC-GLORE framework enables secure collaboration across multiple parties with high computation precision under different settings.

In addition, Table 4 shows the performance differences in terms of OT time, circuit computing time and total time for 2-party, 3-party and 4-party setups. We can see that, for different setups, the total number of AND gates are roughly the same. However, the OT and total time costs increase significantly as the number of parties increases. This is because the GWM project requires pairwise OT communication among parties. The circuit computing time is similar between 2-party and 3-party setups under the *local* settings. However, the circuit computing time becomes much larger under the 4-party *remote* setting. This is due to the fact that the servers in the *remote* setting are located under different networks, while the servers in the *local* setting are under the same network. Therefore, the communication cost in remote setting becomes more significant in cross-network setups.

In Table 5, we performed the comparison of models learned from SMAC-GLORE and ordinary logistic regression, based on four real data sets. All the datasets consist of 3 features. Dataset 1 is the Edinburgh dataset and Datasets 2–4 are genome datasets [50]. We can see that our SMAC-GLORE produced very close results to those produced by ordinary logistic regression.

Limitations and discussion

Our project, which is in part based on the GMW project, developed secure fixed-point algorithms to handle floating number computation and constructed pipelines for securely building a distributed logistic regression model. The experimental results demonstrated the feasibility of the proposed framework, but there are still some limitations.

First, all the variables are represented in a 28-bit fixed-point format, where we allocate 1 bit for the sign (i.e., positive or negative numbers) and 11 bits for fractional part of a floating number. Therefore, the proposed framework cannot handle a number that is larger than 2^{16} or has precision higher than $\frac{1}{2^{11}}$. Although the range and precision of a floating number can be improved by adding more bits in the fixed-point format, it will result in a significant increase of circuit size. Another potential workaround is to replace the fixed-point format with a floating-point format, which will require further investigation. In addition, during fixed-point multiplication, results may end up taking twice the number of bits, where the intermediary product results have 56 bits. In order to maintain a consistent 28-bit fixed-point encoding format, we need to truncate the intermediary product results, which may also result in precision loss. Based on our simulation, we need at least an 80-bit fixed point representation to achieve precision of a single precision floating point encoding format.

Table 3 Model parameters β learned in local 2-party, 3-party scenarios, and remote 4-party scenarios

β	2-party				3-party				4-party (remote)			
	Value	Wald test			Value	Wald test			Value	Wald test		
		SE	Z value	p-value		SE	Z value	p-value		SE	Z value	p-value
β_1	−0.6182	0.7759	−0.7968	0.4256	−0.6182	0.7759	−0.7968	0.4256	−0.6182	0.7759	−0.7968	0.4256
β_2	2.5454	0.8461	3.0084	0.0026	2.5454	0.8461	3.0084	0.0026	2.5454	0.8461	3.0084	0.0026
β_3	1.2246	1.1226	1.0909	0.2753	1.2246	1.1226	1.0909	0.2753	1.2246	1.1226	1.0909	0.2753
β_4	0.6177	0.8283	0.7457	0.4558	0.6177	0.8283	0.7457	0.4558	0.6177	0.8283	0.7457	0.4558

Table 4 Computing performances in local 2-party, 3-party scenarios, and remote 4-party scenarios

	2-party		3-party			4-party (remote)			
	Party 1	Party 2	Party 1	Party 2	Party 3	Party 1	Party 2	Party 3	Party 4
OT time (s)	1,785.51	4,021.83	5,075.79	5,192.17	5,168.24	7,225.62	15,927.10	10,051.50	9,929.10
Computing time (s)	601.76	625.59	538.79	659.43	612.51	650.65	1288.07	1011.14	857.72
Total time (s)	2,607.11	4,683.56	5,822.17	6,064.08	5,976.22	8123.55	17,556.40	11,412.60	11,026.5
# of AND gates	355,074,784		355,075,216			355,075,648			

Second, to handle secure distributed exponential and logit function evaluations, we implemented a secure Taylor expansion algorithm up to a 15 order and truncated the input range (i.e., −5 to 5). The output precision of both functions can be further improved by increasing the order of expansion and of the truncated input range, but this results in additional computational costs. Similarly, we also developed a secure matrix inversion protocol based on an iterative algorithm [41], which only requires multiplication and addition operations. This secure matrix inversion protocol may also introduce precision loss in the final results.

Third, the current implication of the secure multiplication primitive is partially based on the Strassen algorithm, which requires several levels of block-wise decompositions of the input matrix to achieve maximum performance gain. However, we only perform a one-level decomposition rather than repeating the procedure in the whole multiplication. As matrix multiplication is one of the most time-consuming operations, we may need to fully utilize the Strassen algorithm to improve the performance in our future work.

In addition, based on our experiments, we find that the OT phase contributes the most to the computational time, due to the limitations of the GMW project which uses only a single thread for each pair's OT procedure. To reduce the time for OT and improve performance, we can resort to parallel computation in the OT procedure. The GMW project was developed for a 32-bit computing environment, which means that the total number of gates cannot exceed 2^{31}. Therefore, in our future work, we plan to extend GMW to support a 64-bit address space, in which we can handle larger data sets. Another limitation is that the GMW project needs to preload the entire circuit into memory during the computation, which requires a very large amount of memory for a complex circuit. In our future work, we will investigate the possibility of dynamically generating a part of a circuit that will be required for the next execution to reduce memory consumption.

Although the proposed framework can protect the entire model learning phase, there is still no protection of the final output of the learned model parameter. Differential privacy has emerged as one of the strongest privacy guarantees for statistical data release [51]. Roughly speaking, it ensures (to a pre-defined extent, and quantifiably) that no risk is incurred when data from individual patients are entered in a particular database. It will be useful to integrate an optional component to enable differentially private model learning [52] in our future work.

Conclusion

In this study, we developed a secret-sharing, circuit-based SMAC-GLORE framework. To overcome the limitation of GMW, which only supports integer operations, we designed a fixed-point encoding format to support floating number arithmetic in SMAC-GLORE. We also implemented several secure matrix-operation primitives and built a pipeline for secure distributed logistic regression calculation. SMAC-GLORE is able to build a shared model without sharing each party's private data. To the best of the authors' knowledge, the proposed SMAC-GLORE is the first attempt to enable secret-sharing circuit based secure distributed logistic regression model learning for biomedical research studies. The experimental results show that our framework is reliable and can be used to solve practical problems in secure distributed logistic regression model learning.

Table 5 Differences between models learned from SMAC-GLORE and Ordinary Logistic Regression (LR) for Datasets 1-5

β	Dataset 1		Dataset 2		Dataset 3		Dataset 4	
	2-party SMAC-GLORE	Ordinary LR	2-party SMAC-GLORE	Ordinary LR	2-party SMAC-GLORE	Ordinary LR	2-party SMAC-GLORE	Ordinary LR
β_1	1.7632	1.7647	−0.6592	−0.6567	−0.5093	−0.5066	−1.5126	−1.5168
β_2	0.3369	0.3374	0.3174	0.3179	0.5767	0.5777	−0.3516	−0.3488
β_3	1.1885	1.1902	−0.2212	−0.2195	0.4102	0.4138	0.2822	0.2855
β_4	−1.6514	−1.6500	−1.3115	−1.3098	−1.8940	−1.8939	−1.4873	−1.4873

Abbreviations

CSP, computation service provider; CRYSP, crypto service provider; GMW, Goldreich, Micali, and Wigderson; MSE, mean squared error; OT, oblivious transfer; SMC, secure multi-party computation; SMAC-GLORE, secure multi-pArty computation grid logistic regression

Declaration

Publication of this article has been funded by the NIH grants R00HG008175, K99HG008175, R01HG007078, R00LM011392, R21LM012060, and U54HL108460. This article has been published as part of BMC Medical Informatics and Decision Making Volume 16 Supplement 3, 2016. Selected articles from the 5th Translational Bioinformatics Conference (TBC 2015): medical genomics. The full contents of the supplement are available online https://bmcmedgenomics.biomedcentral.com/articles/supplements/volume-16-supplement-3.

Availability of data and materials

http://research.ucsd-dbmi.org/SMAC_GLORE.

Authors' contributions

The authors HS, CJ, and SW contributed the majority of the writing and conducted major parts of the methodology and experiments. WD wrote a part of this paper and provided detailed edits. YT and XJ provided helpful comments on both methods and presentation. LO-M provided the motivation for this work, detailed edits and critical suggestions. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details

[1]Department of Biomedical Informatics, University of California, San Diego, CA 92093, USA. [2]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13210, USA. [3]School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK 74135, USA.

Published: 25 July 2016

References

1. Ohno-Machado L, Agha Z, Bell DS, Dahm L, Day ME, Doctor JN, et al. pSCANNER: patient-centered Scalable National Network for Effectiveness Research. J Am Med Inform Assoc. 2014;21(4):621–6.
2. Schilling LM, Kwan BM, Drolshagen CT, Hosokawa PW, Brandt E. Scalable Architecture for Federated Translational Inquiries Network (SAFTINet) Technology Infrastructure for a Distributed Data Network. eGEMs. 2013;1(1):1–13.
3. McCarty CA, Chisholm RL, Chute CG, Kullo IJ, Jarvik GP, Larson EB, et al. The eMERGE Network: a consortium of biorepositories linked to electronic medical records data for conducting genomic studies. BMC Med Genomics. 2011;4(1):1–13.
4. Naveed M, Ayday E, Clayton EW, Fellay J, Gunter CA, Hubaux J-P, Malin BA, Wang X. Privacy and Security in the Genomic Era; 2014. arXiv preprint arXiv: 1405.1891.
5. Vaidya J, Shafiq B, Jiang X, Ohno-Machado L. Identifying inference attacks against healthcare data repositories. AMIA Summits Transl Sci Proc. 2013;2013:262–6.
6. Sweeney L, Abu A, Winn J. Identifying Participants in the Personal Genome Project by Name (A Re-identification Experiment); 2013. arXiv preprint arXiv: 1304.7605.
7. Gymrek M, McGuire AL, Golan D, Halperin E, Erlich Y. Identifying personal genomes by surname inference. Science. 2013;339(6117):321–4.
8. Wang R, Li YF, Wang X, Tang H, Zhou X. Learning your identity and disease from research papers. In: Proceedings of the 16th ACM conference on Computer and communications security - CCS'09. 2009. p. 534–44.
9. Erlich Y, Narayanan A. Routes for breaching and protecting genetic privacy. Nat Rev Genet. 2014;15(6):409–21.
10. Homer N, Szelinger S, Redman M, Duggan D, Tembe W, Muehling J, et al. "Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays,". PLoS Genet. 2008;4(8):e1000167.
11. Wu Y, Jiang X, Kim J, Ohno-Machado L. Grid Binary LOgistic REgression (GLORE): building shared models without sharing data. J Am Med Inform Assoc. 2012;2012(5):758–64.
12. Lu C-L, Wang S, Ji Z, Wu Y, Xiong L, Jiang X, Ohno-Machado L, Li X, Jiang X, Ohno-Machado L, Xiong L, Jiang X, Ohno-Machado L. WebDISCO: A web service for distributed cox model learning without patient-level data sharing. J Am Med Inform Assoc. The Oxford University Press; 2015;22(6):1212-1219.
13. Wu Y, Jiang X, Wang S, Jiang W, Li P, Ohno-Machado L. Grid multi-category response logistic models. BMC Med Inform Decis Mak. 2015;15(1):1–10.
14. Jiang W, Li P, Wang S, Wu Y, Xue M, Ohno-Machado L, Jiang X. WebGLORE: a web service for Grid LOgistic REgression. Bioinformatics. 2013;29(24):3238–40.
15. El Emam K, Samet S, Arbuckle L, Tamblyn R, Earle C, Kantarcioglu M. A secure distributed logistic regression protocol for the detection of rare adverse drug events. J Am Med Inform Assoc. 2013;20(3):453–61.
16. Slavkovic AB, Nardi Y, Tibbits MM. "Secure" Logistic Regression of Horizontally and Vertically Partitioned Distributed Databases. Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on. IEEE; 2007. pp. 723-728.
17. Li Y, Jiang X, Wang S, Xiong H, Ohno-Machado L. "VERTIcal Grid lOgistic regression (VERTIGO)". J Am Med Inform Assoc. 2016;23(3):570–9.
18. Fienberg S, Fulp W, Slavkovic A, Wrobel T. Secure Log-Linear and Logistic Regression Analysis of Distributed Databases. In: Privacy in Statistical Databases. 2006. p. 277–90.
19. Wang S, Jiang X, Wu Y, Cui L, Cheng S, Ohno-Machado L. EXpectation Propagation LOgistic REgRession (EXPLORER): Distributed Privacy-Preserving Online Model Learning. J Biomed Inform. 2013;46(3):1–50.
20. Yu S, Fung G, Rosales R, Krishnan S, Rao RB, Dehing-Oberije C, Lambin P. Privacy-preserving cox regression for survival analysis. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008. p. 1034–42.
21. O'Keefe CM, Sparks RS, McAullay D, Loong B. Confidentialising Survival Analysis Output in a Remote Data Access System. J Priv Confidentiality. 2012;4(1):6.
22. Lu C-L, Wang S, Ji Z, Wu Y, Xiong L, Jiang X, Ohno-Machado L. WebDISCO: a Web service for DIStributed COx model learning without patient-level data sharing. In: Translational Bioinformatics Conference. 2014.
23. Du W, Han YS, Chen S. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: Proceedings of the 4th SIAM International Conference on Data Mining. 2004. p. 222–33.
24. Wolfson M, Wallace SE, Masca N, Rowe G, Sheehan NA, Ferretti V, et al. DataSHIELD: resolving a conflict in contemporary bioscience performing a pooled analysis of individual-level data without sharing the data. Int J Epidemiol. 2010;39(5):1372–82.
25. Sparks R, Carter C, Donnelly JB, Keefe CMO, Duncan J, Keighley T, McAullay D. Remote access methods for exploratory data analysis and statistical modelling: Privacy-Preserving Analytics. Comput Methods Programs Biomed. 2008;91(3):208–22.
26. Fienberg S, Nardi Y, Slavković A. "Valid statistical analysis for logistic regression with multiple sources. Prot Pers While Prot People. 2009;5661:82–94.
27. Karr AF, Lin X, Sanil AP, Reiter JP. Analysis of integrated data without data integration. Chance. 2004;17(3):26–9.
28. Karr AF, Feng J, Lin X, Sanil AP, Young SS, Reiter JP. Secure analysis of distributed chemical databases without data integration. J Comput Aided Mol Des. 2005;19(9):739–47.
29. Karr AF, Fulp WJ, Vera F, Young SS, Lin X, Reiter JP. Secure, privacy-preserving analysis of distributed databases. Technometrics. 2007;49(3):335–45.
30. Karr AF. Secure Statistical Analysis of Distributed Databases, Emphasizing What We Don't Know. J Priv Confidentiality. 2009;1:197–211.
31. Tang Y, Liu L, Iyengar A, Lee K, Zhang Q. e-PPI: Locator Service in Information Networks with Personalized Privacy Preservation. In: IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014. 2014. p. 186–97.
32. Tang Y, Liu L. Privacy-preserving multi-keyword search in information networks. IEEE Trans Knowl Data Eng. IEEE; 2015;27(9):2424–2437.
33. Tang Y, Wang T, Liu L, Meng S, Palanisamy B. Privacy preserving indexing for ehealth information networks. In: Proceedings of the 20th ACM

international conference on Information and knowledge management. 2011. p. 905–14.
34. Wang S, Zhang Y, Dai W, Lauter K, Kim M, Tang Y, Xiong H, Jiang X. HEALER: Homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS. Bioinformatics. Oxford Univ Press. 2016; 32(2):211–218.
35. Zhang Y, Dai W, Wang S, Kim M, Lauter K, Sakuma J, et al. SECRET: Secure Edit-distance Computation over homomoRphic Encrypted daTa. In: 5th Annual Translational Bioinformatics Conference (accepted). 2015.
36. Zhang Y, Dai W, Jiang X, Xiong H Wang S. FORESEE: Fully Outsourced secuRe gEnome Study basEd on homomorphic Encryption. BMC Med Inform Decis Mak. BioMed Central Ltd; 2015;15(Suppl5):S5.
37. Yao AC. Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). 1982. p. 160–4.
38. Lindell Y, Pinkas B. A proof of security of yao's protocol for two-party computation. J Cryptol. 2009;22(2):161–88.
39. Bellare M, Hoang VT, Rogaway P. Foundations of Garbled Circuits. In: Proceedings of the 2012 ACM conference on Computer and communications security - CCS'12. 2012. p. 784–96.
40. Chen F, Wang S, Mohammed N, Cheng S, Jiang X. PRECISE:PRivacy-prEserving Cloud-assisted quality Improvement Service in hEalthcare. In: Translational Bioinformatics Conference. 2014.
41. G. Boolos, J. P. Burgess, and R. C. Jeffrey, Computability and logic. Cambridge: Cambridge university press; 2002.
42. Golić JD. Cryptanalysis of alleged A5 stream cipher. In: Advances in Cryptology—EUROCRYPT'97. 1997. p. 239–55.
43. Even S, Goldreich O, Lempel A. A randomized protocol for signing contracts. Commun ACM. 1985;28(6):637–47.
44. S. G. Choi, K.-W. Hwang, J. Katz, T. Malkin, and D. Rubenstein, "Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces," in *Topics in Cryptology–CT-RSA 2012*. Berlin Heidelberg: Springer; 2012. pp. 416–432.
45. Goldreich O, Micali S, Wigderson A. How to play any mental game - a completeness theorem for protocols with honest majority. In: Proceedings of the 19th ACM Symposium on Theory of Computing (STOC). 1987. p. 218–29.
46. Naor M, Pinkas B. Computationally secure oblivious transfer. J Cryptol. 2005;18(1):1–35.
47. J. Von Neumann, "First Draft of a Report on the EDVAC," in *The Origins of Digital Computers*. Berlin Heidelberg: Springer; 1982. pp. 383–392.
48. Böhning D. The lower bound method in probit regression. Comput Stat Data Anal. 1999;30(1):13–7.
49. Nardi Y, Fienberg SE, Hall RJ. Achieving both valid and secure logistic regression analysis on aggregated data from different private sources. J Priv Confidentiality. 2012;4(1):9.
50. "Sample genotype data for analysis of case–control studies in in genetic epidemiology." [Online]. Available: http://artax.karlin.mff.cuni.cz/r-help/library/CGEN/html/SNPdata.html. [Accessed: 01 Sept 2015].
51. Dwork C. Differential Privacy: A Survey of Results. In: Theory and Applications of Models of Computation. 2008. p. 1–19.
52. Ji Z, Jiang X, Wang S, Xiong L, Ohno-Machado L. Differentially private distributed logistic regression using private and public data. BMC Med Genomics. 2014;7 Suppl 1:S14.