Computer Communications 153 (2020) 208–216

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Privacy preserving distributed data mining based on secure multi-party computation

Jun Liu [a,*], Yuan Tian [a], Yu Zhou [a], Yang Xiao [a], Nirwan Ansari [b]

[a] Center for Data Science, Beijing University of Posts and Telecommunications, Beijing, China
[b] Advanced Networking Lab., Electrical and Computer Engineering Department, New Jersey Institute of Technology, NJ, United States

ARTICLE INFO

Keywords:
Secure data mining
Multi-party computation
SPDZ protocol
Secret sharing
Matrix optimization
Privacy preserving computing

ABSTRACT

Data mining is an important task to understand the valuable information for making correct decisions. Technologies for mining self-owned data of a party are rather mature. However, how to perform distributed data mining to obtain information from data owned by multiple parties without privacy leakage remains a big challenge. While secure multi-party computation (MPC) may potentially address this challenge, several issues have to be overcome for practical realizations. In this paper, we point out two unsupported tasks of MPC that are common in the real-world. Towards this end, we design algorithms based on optimized matrix computation with one-hot encoding and LU decomposition to support these requirements in the MPC context. In addition, we implement them based on a SPDZ protocol, a computation framework of MPC. The experimental evaluation results show that our design and implementation are feasible and effective for privacy preserving distributed data mining.

1. Introduction

In recent years, we have witnessed an unprecedented big data explosion than ever before, especially for private data in business environments. For most companies and organizations that own private big data, such as electronic commerce, online-banking and cross-field research data, the data infrastructure is built on a nationwide or worldwide distributed system. In such a distributed environments using vulnerable diversified communication channels, private data or privacy information leakage caused by network deception attacks or eavesdropping tricks presents new challenges for data mining [1]. Therefore, enhanced privacy preserving data mining methods are in urgent need for secure and reliable information sharing.

Sachan et al. [2] introduced a number of privacy preservation methods for data mining, such as K-anonymity, classification, clustering, association rule, distributed privacy preservation, randomization, taxonomy tree and condensation methods. In general, these different privacy preserving data mining methods protect privacy in a common way, which is changing sensitive data to mask or erase the original sensitive information to be concealed. The goal is to achieve a trade-off between accuracy and privacy protection in a data mining task. These methods could be further classified into five categories, which are anonymization based, randomized response based, perturbation based, condensation based and cryptography based privacy preserving data mining methods [1–3]. Unfortunately, the usages of anonymization based and randomized response based methods are limited because of huge information loss. The perturbation based and condensation based methods are performing relatively poorly with respect to privacy qualification, which is a metric used to evaluate how closely the origin value of an attribute can be estimated [3]. Cryptography based methods apply cryptographic techniques to carry out data mining tasks. They try not to reveal any information on original data except that can be inferred from the task output [4]. Secure multi-party computation (MPC) is a typical cryptography based method. In recent years, MPC has been commonly used to achieve privacy preserving distributed data mining because it is more effective and efficient than other methods. The secure multi-party computation problem refers to jointly computing a statistical function together by multiple parties. After the computation, each party can acquire the correct results and no one can get more knowledge than the data inferred from the public results. For data in real scenarios, MPC can support fixed-point and floating-point operations [5,6], and the arithmetic operations can be implemented with controlled linear complexity [7,8]. Owing to these advantages, exploring MPC for privacy preserving distributed data mining has attracted much attention in recent years. However, there are still some problems to be tackled for practical applications.

In this paper, we focus on two most common distributed data mining tasks that are not supported in MPC, namely, calculating statistics of different types of data and studying the relationship between variables

* Corresponding author.
E-mail addresses: liujun@bupt.edu.cn (J. Liu), tianyuan@bupt.edu.cn (Y. Tian), zyicy@bupt.edu.cn (Y. Zhou), zackxy@bupt.edu.cn (Y. Xiao), nirwan.ansari@njit.edu (N. Ansari).

https://doi.org/10.1016/j.comcom.2020.02.014
Received 25 July 2019; Received in revised form 19 December 2019; Accepted 3 February 2020
Available online 8 February 2020

by linear regression. To support these requirements, we design algorithms based on optimized matrix computation with one-hot encoding and LU decomposition [9]. We utilize MP-SPDZ [10], which is a cross-platform multi-party computation programming framework based on the popular MPC protocol stack named Multi-Protocol SPDZ, as the base framework to implement our solutions. The design and implementation are optimized to achieve high communication performance and low computational costs. This paper is an elaborated and extended version of a conference paper to be presented at the 11th International Conference on Advanced Infocomm Technology [11]. The key enhancements of this version includes: (1) we implement the QR decomposition for matrix operation in the SPDZ framework with Householder Reflection, (2) we compare and analyze the performance of QR and LU decomposition for different data size in real scenarios, (3) we perform an extra comprehensive mining task on vehicle driving logs and evaluate its preciseness and efficiency, (4) we further optimize data flow in task 1 by extracting the plain-text encoding out of the SPDZ protocol, and (5) we compare the proposed algorithm and implementations with existing methods by experiments. In summary, the main contributions of this paper are as follows:

- We introduce two real computation scenarios in which current secure multi-party computation method cannot fully protect privacies. On the basis of variable encoding, data vectorization and matrix operation, we propose and implement a distributed data mining solution to ensure private data secure, covering basic arithmetics and some statistical operations as well.
- Our design and high performance algorithm can be applied to the regression problem like the least squares method. We compare the performance of two possible methods, namely, QR decomposition and LU decomposition. With less time-consuming multiplication operations and more numerical stability, the LU decomposition method is chosen. Then, we implement the high performance regression algorithm in the MPC context.
- We carry out a number for experiments with real-world data to verify the effectiveness and efficiency of our design and implementation on the MP-SPDZ framework. We record key performance metrics, which are bytecode size, transmission data size, preciseness and execution time. The comparison of these metrics with traditional privacy preserving methods demonstrate that our method is feasible and effective for privacy preserving distributed data mining.

The rest of this paper is structured as follows. In Section 2, we briefly introduce related works of MPC and its applications. Then, we define the problems that have not been solved in current MPC implementations. We also detail our solution in this section. In Section 4, we present the experimental results of our design and implementation to show their effectiveness and performance. In Section 5, we conclude and point out future works.

2. Related work

Acquiring information by gathering data from multi-parties is an important task for data mining. In a business environment, preventing privacy leakage while carrying out this task is a critical requirement. Towards this end, Yao [12] firstly and formally introduced the secure two-party computation (2PC) problem and its solution in 1982. Then, the 2PC case was generalized and extended to the multi-party computing (MPC) problem by Goldreich et al. [13,14]. After that, a number of researchers started to explore practical ways to enable MPC. The first big breakthrough emerged when Gentry [15] designed a homomorphic encryption with an ideal lattices scheme. Then, a number of researchers proposed various MPC implementations, such as Semi-homomorphic Encryption [16], Lost-cost Multiparty Computation for Dishonest Majority [17] and Active-Secure Two-Party Computation [18]. Most of these methods can be categorized into two types of approaches, homographic encryption and secret sharing. Gentry [15] realized the first scheme with both additive and multiplicative homomorphism. However, if MPC utilizes finite homographic encryption, it will take long time to calculate complex circuits when performing inevitable noise elimination. On the contrary, secret sharing method [16] can calculate infinite times of any addition and multiplication with additional data exchange. Therefore, the homomorphic encryption based approach is not competitive in practice as compared to the secret sharing based approach.

Based on the concept and theory of MPC and secret sharing, a group of researchers designed and implemented a computation framework and protocol named SPDZ [19]. SPDZ can realize arbitrary combination of addition and multiplication operations through secret sharing and homomorphic encryption. SPDZ achieves $O(n)$ storage space complexity, public-key operations less than $O(n^2)$ and support for a larger number of fields. In this paper, we take SPDZ as the basic framework and make extensions on optimized matrix operations and linear regression solution.

There have been some initial research works on data mining focused on the applications of MPC. The study in [20] indicates that most ordinary forms of multi-input computation could be transformed into MPC problems. Dan et al. [21] utilized the SHAREMIND framework and SECREC pseudo-language to realize MPC and secret-sharing scheme for safe financial data analysis. They used JavaScript data-distributing scripts running on client-side web frontend. In their design, the privacy data transmitted on web service is not end-to-end. It leaves no direct access to the miner host for the data provider. In addition, the shared data depends on the number of miner hosts rather than that of data providers. Nikolaenko et al. [22] proposed and implemented a system for privacy-preserving ridge regression algorithm. They employed a hybrid approach that used both homomorphic encryption and Yao garbled circuits methods to learn ridge regression coefficients for a large number of users. As the initial research work, it proves the feasibility of privacy-preserving data mining based on MPC. However, its performance is not good enough for practical applications. Hall et al. [23] proposed to use homomorphic encryption for solving the problem of linear regression where the data are split up and held by different parties. This work focus on fixed-point representation of real numbers with scale factor and its calculations, though lack of describing how to get product of two encrypted value in Paillier's scheme and introducing more complexity from more parameters. Dankar et al. [24] proposed and implemented an algorithm for computing the privacy preserving linear regression model based on its sequentially multiplication of each party. However, it did not describe how to achieve multiplication of two encrypted number in the condition of homomorphic encryption. Authors in [25] proposed an information-theoretically secure protocol for securely computing a linear regression model. The protocol involves a computationally heavy offline trusted initialization phase and a light online computation phase, which makes it practical for computing a large number of data. Gascon et al. [26] proposed privacy-preserving protocols for computing linear regression models mainly on Yao's garbled circuits and oblivious transfer (OT) protocol. This work is an improvement of [22] for high-dimensional data with one million records and one hundred features. Giacomelli et al. [27] proposed a novel privacy-preserving mining system that can train a ridge linear regression model using only linearly-homomorphic encryption method. The system is more efficient than [22] by removing Yao's protocol, which is the main bottleneck in [22]. Our work is inspired by these works to empower MPC for privacy preserving distributed data mining.

3. Problem definition and solution

In this section, we will first introduce two real-world tasks of privacy preserving distributed mining. Then, we will illustrate the traditional way to solve these tasks by secret sharing, and point out its potential risk of privacy leakage. Finally, we will describe our solutions for these two tasks.

3.1. Problem definition

We define two problems to be solved in this paper by introducing two real-world privacy preserving distributed mining tasks.

Task 1: Calculating Statistics of Different Types of Data

Calculating statistical indicators is a common task for business companies. In one of our engaged data mining projects, we are required to perform a task of calculating statistics of different types of data. There are a number of business companies that own trading records of different types of goods, say, cotton. Each type of cotton is marked with a code, such as $3128B$ and $4128B$. Owing to varied quality and market requirements, each trading record of a given batch of cotton has a specific price, which produces a tuple of $\langle code, price\rangle$. All companies are interested in the average price of each code and its variance for their market planning and business strategies. They have to gather all trading records together to calculate the average price of each type of cotton. However, trading records are critical business secrets for every company. They definitely do not want to provide the original trading data, even though they all care about the average prices. Obviously, it is a typical privacy preserving distributed data mining task. Although we can solve it by the traditional MPC-based method, it presents a risk of privacy leakage for cotton type related information. We will show the risk in the next section. Here, we first define the problem of calculating statistics of different types of data under privacy preserving distributed data mining condition as:

- Given data in the form of $\langle type, value\rangle$ tuples, the problem is to calculate the statistical indicators of given data, such as mean, variance, maximum and minimum values, without privacy leak risk for both $type$ and $value$ in original data.

Task 2: Studying the Relationship between Variables by Linear Regression

Studying the relationship between variables by linear regression is another important data mining task. Here, we take the cotton trading business as an example again. Owing to ever-changing market demands, the price of a given batch of cotton is determined by the quality factors, such as color, fiber length and strength. In this condition, price is the dependent variable and quality factors are independent variables. Just like the average price, the influence of quality factors on the price is also interested by all companies. In general, we can use the linear regression to reveal the relationship between variables. The same concern of business privacy leakage exists if these companies are asked to disclose their trading records to perform linear regression analysis. Hence, we define the problem of performing linear regression under privacy preserving distributed data mining condition as:

- Suppose we have a set of tuples formed as $\langle dep\_var, indep\_var\_1, \ldots, indep\_var\_n\rangle$, in which dep_var and $indep\_var$ refer to dependent variable and independent variable, respectively. All independent variables in a row compose a matrix A. All dependent variables compose a column vector b. The problem is to find an optimal x for minimizing the error of $Ax = b$, without privacy leakage risk for both A and b in the original data.

3.2. Traditional secret sharing based MPC solution

3.2.1. Secret sharing based MPC

Secret sharing refers to a method for distributing a secret amongst a group of participants, in which the secret is split into several shares given to shareholders. Each individual shareholder learns nothing about the secret, but yet the secret can be reconstructed if shares are re-combined together.

We take a field element $a \in A$ as an example to explain two-party secure multi-party computation based on secret sharing. We can split up a randomly and uniformly into two pieces as $a = a_1 + a_2$. Then, we give the value a_1 and a_2 to party P_1 and P_2, respectively. Neither party knows the value a. However, we can see that they can reconstruct a by combining a_1 and a_2 together. In the following part, we will use expression $\langle a\rangle$ to represent that the value a is a secret shared between all parties. For each i, party P_i has a_i and $\sum_i a_i = a$. All data in A are supposed to be represented by elements in a finite field. It is $A \subseteq \mathbb{F}_{\mathbb{Q}}$ where $\mathbf{Q}$ is a large prime number.

In terms of the way for secret sharing, there are a number of complicated methods, such as multiplicative sharing $a = a_1 \cdot a_2$ and Shamir sharing [28]. However, the additive scheme is the most common method for secret sharing in MPC applications. It can be generalized as two algorithms $Share$ and $Reconstruct$ among N parties. They are shown in Algorithms 1 and 2.

3.2.2. SPDZ protocol for secret sharing based MPC

Secret sharing is just a theory for performing MPC. It needs a computation framework for implementing MPC based on secret sharing. SPDZ is such secure multi-party computation protocol based on secret sharing and MACs [19]. In the SPDZ protocol, a MPC task is performed in two phases. The first phase is an off-line preprocessing phase in which numerical triples are created and pre-shared among parties. It is helpful for improving performance and protecting against a dishonest majority by somewhat homomorphic encryption (SHE). The second phase is an online phase to execute actual calculation. With an off-line preprocessing phase for offloading works, the computational cost during actual calculation of online phase is low. The overall process and the basic arithmetic operations of the SPDZ protocol are described as follows:

- Before the preprocessing computation takes place, parties agree on some message authentication codes with which they check all their data. To check the message authentication codes, we

Algorithm 1 $Share_N$

Input:
$a \in \mathbb{F}_{\mathbb{Q}}$ → The input number secret
$N \in \mathbb{Z}^+$ → The number of secret-sharing parties
Output:
$B_{list} = [b_i]_{1 \leqslant i \leqslant N}$ → The output split secret for each party

```
temp = 0
for j = 1 to N − 1 do
    # randomly choose number
    b_i = random_range(Q)
    temp = temp + b_i
end for
b_N = (a − temp) mod Q
return B_list
```

Algorithm 2 $Reconstruct_N$

Input:
$B_{list} = [b_i]_{1 \leqslant i \leqslant N}$ → The secret list got from each party
Output:
$a \in \mathbb{F}_{\mathbb{Q}}$ → The output additive sum

```
sum = 0
for each b_i in B_list do
    # sum all shares
    sum = sum + b_i
end for
a = sum mod Q
return a
```

need a global key α, which is produced from the preprocessing in following form:

$$\|\alpha\| := ((\alpha_1, \ldots, \alpha_n), (\beta_i, \gamma(\alpha)_1^i, \ldots, \gamma(\alpha)_n^i)_{i=1,\ldots,n})$$

where $\alpha = \sum_i \alpha_i$ and $\sum_j \gamma(\alpha)_i^j = \alpha\beta_i$. Party P_i holds $[\alpha_i, \beta_i, \gamma(\alpha)_1^i, \ldots, \gamma(\alpha)_n^i]$. Computation of $\gamma(\alpha)_i \leftarrow \sum_j \gamma(\alpha)_i^j$ is the message authentication process that verifies α under P_i's private key β_i. To open $\|\alpha\|$, each P_j sends to all P_i its share α_j of α and $\gamma(\alpha)_i^j$ of the message authentication code on α made by P_i's private key. After that, P_i checks if $\sum_j \gamma(\alpha)_i^j = \alpha\beta_i$.

- The data shared in SPDZ use the additive scheme defined before. Each shared value a is represented in the following way:

$$\langle a \rangle := (\delta, (a_1, \ldots, a_n), (\gamma(\alpha)_1, \ldots, \gamma(\alpha)_n))$$

where $a = \sum_{i=1}^n a_i$, $\alpha(a+\delta) = \sum_{i=1}^n \gamma(\alpha)_i$ and δ is a public value. The interpretation is that $\gamma(\alpha) \leftarrow \sum_{i=1}^n \gamma(\alpha)_i$ is the MAC authenticating a with the global key α.
- In the preprocessing phase, the protocol generates the global key $\|\alpha\|$, random number pairs $\|\mathbf{r}\|, \langle \mathbf{r} \rangle$ and multiplicative triples $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ satisfying $c = a \cdot b$. In this process, technologies of Reshare, sacrificing, Zero-Knowledge Proofs of Plaintext Knowledge (ZKPoPKs) and SHE are used.
- In the online computation phase, the first task is performing secret sharing. To share P_i's input x_i, P_i takes an available pair $\|\mathbf{r}\|$ and $\langle \mathbf{r} \rangle$. Then, $\|\mathbf{r}\|$ is opened to P_i. After that, P_i broadcasts $\epsilon \leftarrow x_i - r$ and every party computes $\langle x_i \rangle \leftarrow \langle \mathbf{r} \rangle + \epsilon$. We can perform arithmetic operations as below.
  $\widetilde{+}$: **addition** with $\langle x \rangle, \langle y \rangle$. Each party locally computes $\langle x \rangle + \langle y \rangle$.
  $\widetilde{\times}$: **multiplication** with $\langle x \rangle, \langle y \rangle$. Parties partially merely open share value without message authentication code $\langle x \rangle - \langle a \rangle$ to get ϵ. They also open $\langle y \rangle - \langle b \rangle$ to get ρ and compute $\langle z \rangle \leftarrow \langle c \rangle + \epsilon\langle b \rangle + \delta\langle a \rangle + \epsilon\rho$. Note that the sum $\langle z \rangle$ is correct because $\sum_i z_i = c + (x-a)b + (y-b)a + (x-a)(y-b) = xy$.
  $\widetilde{-}$: **subtraction** can be easily derived by addition $\widetilde{+}$ with the opposite number.
  $\widetilde{\div}$: **division** can be derived by **multiplication** $\widetilde{\times}$ and techniques from [6] and [8].
- At the end of online phase, SPDZ protocol checks the correctness by verifying the message authentication code values. Supposing $\langle a_1 \rangle, \ldots, \langle a_T \rangle$ be all values publicly opened so far, a random value from preprocessing $\|e\|$ is opened. Each party compute $a \leftarrow \sum_{j=1}^T a_j \cdot e^j$. They also compute their share of the message authentication code by $\gamma_i \leftarrow \sum_{j=1}^T \gamma(a_j)_i \cdot e^j$. After $\|\alpha\|$ is opened, all parties check whether $\alpha \cdot (a + \sum_{j=1}^T \gamma(a_j)_i \cdot e^j) = \sum_i \gamma_i$. If it is correct, the output can be opened to get results. Otherwise, the protocol aborts.

3.2.3. Performing data mining tasks based on SPDZ

Based on SPDZ, we can achieve the first data mining task, calculating the average value of different types of data. While most of operations in SPDZ are component-wise operations, the more effective way to perform large-scale arithmetics is to compose data into a matrix for computation. Thus, we can make use of vectorization to optimize computation performance. A simple yet useful algorithm to calculate the weighted average value of data in different types is presented in Algorithm 3, in which each party has to provide all private data of each type separately. In the computation process in SPDZ, it will unintentionally discloses the information about the amount of data of each type, which is also sensitive business information for a company.

In the original SPDZ protocol, we cannot finish the second data mining task, studying the relationship between variables by linear regression. To solve the linear regression, we need to use least squares method. However, SPDZ only defines basic arithmetic operations of $\widetilde{+}$ and $\widetilde{\times}$ between secret-shared numbers. It cannot support least squares method directly.

3.3. Our solution

In this section, we will demonstrate how we accomplish the above two tasks by matrix computation with one-hot encoding and LU decomposition, respectively.

3.3.1. Solution for Task 1

As mentioned before, there is a risk of certain type of information leakage if we separately compute the average price of different types of cotton. To solve this problem, we consider to mix up different types of data to avoid revealing the amount of cotton of a certain types. In this way, we can compute statistical indicators by selecting data of each type rather than transfering data of a certain type between involved parties. The key point to achieve this goal is the capability of distinguishing different types in a mixed dataset. To do that, we utilize the one-hot encoding method [29] to combine the type information and data values. An example of one-hot encoding is illustrated in

Algorithm 3 *Weighted_Average* — Algorithm of Computing Weighted Average by MPC

Input:
$W = [w_{ij}] \in \mathbb{R}^{R \times C}$ → The weight matrix with size $R \times C$
$V = [v_{ij}] \in \mathbb{R}^{R \times C}$ → The value matrix with size $R \times C$
Output:
$A = [a_{ij}] \in \mathbb{R}^{1 \times C}$ → The weighted average vector with size $1 \times C$

initialize vectors $wgt\_sum, val\_sum \in \mathbb{R}^{1 \times C}$ with 0s
for each row $\mathbf{w_i}(1 \leqslant i \leqslant n)$ **in** W **do**
  $wgt\_sum = wgt\_sum \widetilde{+} \mathbf{w_i}$
end for
for each row $\mathbf{v_i}(1 \leqslant i \leqslant n)$ **in** V **do**
  $val\_sum = val\_sum \widetilde{+} \mathbf{v_i}$
end for
for $i = 1$ **to** C **do**
  # get average by dividing each weight element-wise
  $A(i) = val\_sum(i) \widetilde{\div} wgt\_sum(i)$
end for
return A

Algorithm 4 *Avg_Var* — Algorithm for Computing Weighted Average and Variance

Input:
$C_{list} = [(type_i, price_i)]_{1 \leqslant i \leqslant 1500}$ → The cotton data collected from each party

initialize T, P with new 1500×10 zero matrices
for each row $\mathbf{p_i}(1 \leqslant i \leqslant 1500)$ **in** P **do**
  #one-hot encoding, 1 for corresponding type
  $\mathbf{p_i}[type_i] = 1$
end for
for each row $\mathbf{t_i}(1 \leqslant i \leqslant 1500)$ **in** T **do**
  $\mathbf{t_i}[type_i] = price_i$
end for
$Avg = Weighted\_Average(T, P)$
for each row $\mathbf{t_i}$ **in** T **do**
  $\mathbf{t_i}[type_i] = (price_i - Avg[type_i])^2$
end for
$Var = Weighted\_Average(T, P)$
return Avg and Var

Algorithm 4. The type information could be encoded into a one-hot vector and so does value information. The one-hot encoding process of the input data is performed by the parties with their own data based on plain coding scripts. It is to encode data types into a weighting matrix consisting of one-hot rows. Data values are also composed as a matrix to be calculated weighted average by column. Noticeably, the encoding process is not running in the SPDZ protocol and that will further improve our computation performance. By this way, only "1" in each row will be counted and different types are able to be distinguished. Algorithm 4 illustrates our algorithm for computing the weighted average and variance of different types of data.

3.3.2. Solution for Task 2

As stated before, the key point to accomplish task 2 is using least square method to solve linear regression. Mathematically, the matrix decomposition method is a good tool for solving least squares problems. However, owing to different implementations of arithmetic operations in SPDZ, we have to carefully consider how to find an efficient way for matrix decomposition in the MPC context. Towards this end, we choose QR [30] and LU [31] decomposition methods as candidates for matrix decomposition. Here, we introduce how we implement QR and LU decomposition for MPC and compare their performance in the experimental evaluation section.

QR Decomposition for MPC

QR decomposition needs to frequently calculate the L2 norm of the given data matrix and call the *sqrt*(square root) operation. We realize it by Newton's method [32]. More specifically, there are three commonly used methods to calculate QR decomposition, Gram–Schmidt, Householder Reflection and Givens Rotation. Since the numerical stability of the Gram–Schmidt process is poor and Givens Rotation needs to compute complicated trigonometric functions, we choose Householder Reflection to realize the QR decomposition. With QR decomposition result $\bar{R}, \bar{Q}$, we can pick front rows from $\bar{R}, \bar{Q}$ to form square matrix R, Q. The least squares solution can be denoted as $R^{-1}Q^T b$, where R is an upper triangular matrix. Its inversion can be accomplished by iteration step by step, which consumes less time of about 4% of the total execution time. To determine whether the error converges in Newton's method, we need to calculate the difference between $(\sqrt{x_{sqrt}})^2$ and x. In the SPDZ protocol, this will induce a certain amount of network communication cost. Moreover, owing to the undetermined convergence rounds for convergence, it will be difficult to estimate the consumption of pre-processing data in advance. In SPDZ, fixed-point numbers representing over 2^{16} are often used. Therefore, if the difference is less than 0.0001, about 10 rounds of communication are needed. In the selected householder QR implementation, the square operation makes up 13% of the total execution time, which exerts unfavorable impact on performance. In addition, it is inevitable to store the difference between estimated values and real ones to determine the termination condition of Newton's iteration. It will consume a large number of computation and storage resources. In our experimental evaluation, we can see that the performance of QR decomposition is not good for solving linear regression in the MPC context.

LU Decomposition for MPC

In linear algebra, LU decomposition factors a matrix as the product of a lower triangular matrix L and upper triangular matrix U. Suppose we want to linearly combine n different types of data x and parameters θ into $h_\theta(x^{(i)}) = \sum_{i=1}^{n} \theta_i x_i$ to approximate target y. If we have m groups of data, the goal is to minimize the cost function $J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(h_\theta(x^{(j)}) - y^{(j)}\right)^2$. In this condition, the data vectorization should be performed as follows. The input matrix is $X = [(x^{(1)}), (x^{(2)}), \ldots, (x^{(m)})]^T$, where all $1 \le i \le m$ satisfy $(x^{(i)}) = [x_1^{(i)}, \ldots, x_m^{(i)}]$. Similarly, the parameter matrix θ and result matrix Y can be denoted as $\theta = [\theta_1, \theta_2, \ldots, \theta_m]^T$ and $Y = [y_1, y_2, \ldots, y_m]^T$, respectively. Then, the cost function can be written in the matrix form as $J(\theta) = \frac{1}{2}\left[(X\theta - Y)^T(X\theta - Y)\right]$. By setting derivative function $\partial J(\theta)/\partial\theta = 0$, we can get $X^T X\theta = X^T Y$. It could also be seen as solving θ from equation $A\theta = B$, where $A = X^T X$ is a square matrix and $B = X^T Y$. Moreover, if A can be decomposed in the LU way as $A = LU$, we can easily solve θ in two steps. At first, solve $T = U\theta$ in $LU\theta = B$. Then, we can solve θ in $U\theta = T$.

Algorithm 5 shows how to recursively calculate the lower triangular matrix L and upper triangular matrix U of a given matrix in SPDZ. The equation can be solved directly by forward and backward substitution without using the Gaussian elimination process. Noticeably, it is mathematically proved that the Schur complement is always full-ranked if

Algorithm 5 *LU_Decomposition* — LU Decomposition Implementation in SPDZ

Input:
$A = [a_{ij}]_{1\leqslant i\leqslant n, 1\leqslant j\leqslant n}$ → The input square matrix with size $n \times n$
Output:
$L = [l_{ij}]_{1\leqslant i\leqslant n, 1\leqslant j\leqslant n}$ → The lower triangular matrix with size $n \times n$
$U = [u_{ij}]_{1\leqslant i\leqslant n, 1\leqslant j\leqslant n}$ → The upper triangular matrix with size $n \times n$

check whether A is full-ranked, otherwise aborts
let L, U be new $n \times n$ matrices
initialize U with 0s below the diagonal, and initialize L with 1s on the diagonal and 0s above the diagonal
for $k = 1$ **to** n **do**
    # determine the pivot
    $u_{kk} = a_{kk}$
    for $i = k + 1$ **to** n **do**
        # calculate kth columns in matrix L
        $l_{ik} = a_{ik} \div u_{kk}$
        # calculate kth rows in matrix U
        $u_{ki} = a_{ki}$
    end for
    for $i = k + 1$ **to** n **do**
        for $j = k + 1$ **to** n **do**
            # update $\mathbf{A'} - \mathbf{vw^T}_{(n-k)\times(n-k)}$ on A
            $a_{ij} = a_{ij} \sim l_{ik} \times u_{kj}$
        end for
    end for
end for
return L and U

Algorithm 6 *PriceReg* — Linear Regression of Cotton Price and Quality factors by LU Decomposition

Input:
$C_{list} = [(color_i, length_i, micronaire_i, price_i)]_{1\leqslant i\leqslant 1500}$ → The cotton data collected from each party

initialize A with new 1500×10 with 1 in first row, 0 in other rows
initialize b with new 1500×1 with 0s
define function **index**(i):
    return the index of type i in one-hot encoding
for each row $\mathbf{a_i}(1 \leqslant i \leqslant 1500)$ **in** A **do**
    $\mathbf{a_i}$[**index**$color_i$)] = 1
    $\mathbf{a_i}$[**index**($length_i$)] = 1
    $\mathbf{a_i}$[**index**($micronaire_i$)] = 1
end for
for $i = 1$ **to** 1500 **do**
    $b[i] = price_i$
end for
#LU decomposition
$L, U = LU\_Decomposition(A^T A)$
$Cof = A^T b L^{-1} U^{-1}$
return Cof

Table 1
Description of cotton dataset.

Variable name	Variable type	Example value
type	categorical	3128B
color	categorical	level 31
length	categorical	level 28
micronaire	categorical	level B2
price	numerical	14 280

Table 2
Description of vehicle dataset.

Variable name	Variable type	Example value
mileage	numerical	9.96
power	numerical	106
fuel	numerical	3.0

A is full-ranked. Therefore, we are able to safely optimize the least squares method and apply it to accelerate data mining. Algorithm 6 illustrates how to solve the linear regression of cotton price and quality factors by LU Decomposition in the MPC context.

4. Experimental evaluation

In this section, we perform a set of experiments to evaluate our design and implementation for preserving distributed data mining based on MPC.

4.1. Experimental environment

We use a Kernel-based Virtual Machine (KVM) with 8 GB memory and 6 virtual CPU cores on Intel Xeon E3. The machine installs CentOS Linux with version 7.2.1511 core. We utilize the official successor projects of SPDZ, MP-SPDZ [10], with the first release on Feb 14, 2019 to execute experiments. Our designed algorithms were implemented by Python based on the SPDZ protocol.

4.2. Experiment datasets and evaluation tasks

We collected two datasets, cotton trading records and vehicle driving logs, to conduct our experiments. Tables 1 and 2 show the variable names, variable types and example values of these two datasets, respectively. Based on these datasets, we performed three data mining tasks for validation and performance evaluation of our design and implementation.

4.2.1. Task 1: Computing average and variance of cotton price

In this task, our goal is to calculate the average price and variance of different types of cotton without privacy leak. The data can be denoted as two-tuples in form of $(type, price)$. $type$ refers to 10 different types of cotton. In this experiment, we use one-hot encoding to represent 10 different types in form of $\underbrace{0\cdots01\cdots0}_{10\text{ bits}}$. $price$ is the real cotton transaction price and ranges from 10,000 to 15,000. There are 3 trading participants that provide 500 tuples for each. The entire data mining process is illustrated in Algorithm 4.

4.2.2. Task 2: Studying how cotton quality factors impact the price

In this task, our goal is to determine the regression coefficients of different quality factors with cotton price by the least squares method in the MPC context. The data can be denoted as four-tuples in form of $(color, length, micronaire, price)$. $color$ refers to one of four different color-type values $\{21, 22, 31, 32\}$. $length \in \{27, 28, 29\}$ refers to the cotton fiber length. $micronaire \in \{C2, C1, B2, B1, A\}$ refers to the cotton quality level. Here, we also use one-hot encoding to represent 10 different types in 3 indexes in the form of $10\cdots101\cdots10$, where the first bit 1 denotes the constant offset, and the following 3, 2, 4 bits denotes $color$, $length$ and $micronaire$, respectively. Each of three trading participants provides 500 tuples. To avoid multicollinearity, we first detect by examining the correlations between each pair of independent quality factors. If two factors are highly correlated (e.g., 21 in $color$, 27 in $length$), we will remove one column of these two factors in the one-hot encoding matrix. The data mining process is described in Algorithm 6.

4.2.3. Task 3: A comprehensive analysis of vehicle driving logs

In this task, our goal is to perform a comprehensive analysis on data produced by IoV (Internet of Vehicles) with secure multi-party computation. Vehicle mileage, motor power and fuel consumption are collected to perform analysis. The details of vehicle data structures are neglected due to its enormous data types and scales. Each of three vehicle manufacturers provides 300 groups of data. The data mining task procedure is shown in Algorithm 7. There are three sub-task in this task. The first and second sub-tasks are computing the minimum, maximum average and variance value of mileage and motor power in driving logs, respectively. The third sub-task is the linear regression of motor power and fuel consumption per kilometer.

4.3. Comparison of QR and LU decomposition for MPC

To compare the performance of our implemented QR and LU decomposition for MPC, we generate seven datasets. In each dataset, every participant has a matrix with 4 columns and N rows. N varies from 4 to 10 for these seven datasets. We execute the implemented QR and LU decompositions on these datasets. The results of execution time are shown in Fig. 1. We can see that the execution time of LU is relatively the same for all data sizes. However, the execution time of QR show a significant upward trend when the data size increases from 5 to 10. For all data sizes, the performance of LU is better than QR. Based on the theoretical analysis in Section 3.3 and experimental evaluation, we choose LU decomposition for linear regression in the following experiments.

4.4. Validity evaluation

To validate the effectiveness of our design and implementation, we perform the above tasks on both single machine and MPC distributed

Algorithm 7 $VoTDataTest$ — A Comprehensive Analysis of Vehicle Driving Logs

Input:
$M_{list} = [mileage_i]_{1\leqslant i\leqslant 300}$ → The mileage data collected from each party
$P_{list} = [power_i]_{1\leqslant i\leqslant 300}$ → The motor power data collected from each party
$F_{list} = [fuel_i]_{1\leqslant i\leqslant 300}$ → The fuel consumption per kilometer data collected from each party

define function **stat**(L):
    return the statistics (min, max, avg, var) of a list L
Task 3_1:
$m = \textbf{stat}(M_{list})$
Task 3_2:
$p = \textbf{stat}(P_{list})$
Task 3_3:
#concatenate a row of 1s on P_{list} to represent offset
$P = [P_{list}^T, \mathbf{1}_{300}]$
#LU decomposition
$Cof = LU\_Decomposition(P^T P)$
$Cof = P^T F L^{-1} U^{-1}$
return m, p, Cof

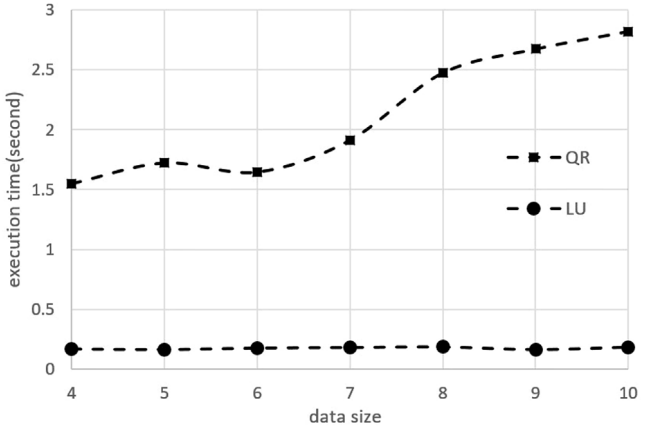

Fig. 1. The execution time of QR and LU decomposition.

Table 3
Validity evaluation of task 1.

Type	avg	avg of MPC	var	var of MPC
1	12451.2	12451.2	83842.6	83842.6
2	11996.2	11996.2	72720.4	72720.4
3	12212.3	12212.3	77324.2	77324.2
4	13547.7	13547.7	38840.9	38840.9
5	13243.1	13243.1	58324.2	58324.2
6	13626.2	13626.2	36740.0	36740.0
7	14346.1	14346.1	32812.2	32812.2
8	12106.2	12106.2	74415.2	74415.2
9	13803.0	13803.0	38212.3	38212.3
10	12231.2	12231.2	79236.6	79236.6

Table 4
Validity evaluation of task 2.

Type	Synthesis data		Cotton data	
	par	par of MPC	par	par of MPC
1	0.193906	0.193906	12943.4	12943.4
2	10.039	10.039	540.334	540.332
3	19.5221	19.5221	201.64	201.64
4	29.8581	29.8581	−218.473	−218.473
5	40.4993	40.4993	283.298	283.298
6	50.0950	50.0948	−140.925	−140.926
7	59.8919	59.8920	478.289	478.288
8	69.9046	69.9045	67.5113	67.5113
9	79.7971	79.7970	−130.823	−130.823
10	89.7018	89.7018	−376.981	−376.982

Table 5
Validity evaluation of task 3.

Task	Maximum relative error
Task 3_1	0.00413%
Task 3_2	0.00126%
Task 3_3	0.0178%

computing cluster. Tables 3–5 shows the results of task 1, 2, 3, respectively. From Tables 3 and 4, we can see that the computation results of single machine and MPC are almost the same. The difference can be almost ignored. In terms of task 3, we compute the maximum relative error of MPC as compared to single machine. All relative errors are very small. The evaluation results of these three tasks prove the effectiveness of our design and implementation.

4.5. Performance evaluation

4.5.1. The overall performance

We gather a set of important metrics to evaluate the overall performance of our design and implementation for the above three tasks. For task 1 and 2, we evaluate the storage cost by the size of transmission data, compiled bytecode and preprocess data volume. The numbers of these metrics in Tables 6 and 7 demonstrate that the storage cost of our design and implementation is acceptable. In addition, computation time of task 1 and 2 are 0.342 s and 0.861 s, respectively. It shows that our solution is suitable for this kind of off-line data mining tasks. From Table 8, we can get the same conclusion for the evaluation of task 3.

Table 6
Overall performance metrics of task 1.

Performance metrics	Performance data
Transmission data	0.596728 MB
Compiled bytecode	3.6 MB
Preprocess data volume	15, 200 bits+4250 triples
Computation time	0.342 s

Table 7
Overall performance metrics of task 2.

Performance metrics	Performance data
Transmission data	12.2973 MB
Compiled bytecode	9.2 MB
Preprocess data volume	141, 800 bits+188, 860 triples
Computation time	0.861 s

4.5.2. The performance with data size increasing

To evaluate if our design and implementation is scalable when the data size increases, we prepared five datasets of cotton trading. For each trading participants, the number of trading records varies from 100 to 500. Then, we conduct task 1 and 2 on these five datasets. Figs. 2 and 3 show changes of the amount of transmission data and the execution time, respectively. From Fig. 2, we can see that the amount of transmission data increases linearly but with different growth rates in task 1 and 2. Task 1 contains 63 KB, 1500 lines of text in the original data, while the actual amount of transmission data amount is 597 KB. It shows that most proportion of transmission lies in the pre-processed data. Task 2 contains 36 KB, 1500 rows of the original data, while the actual amount of transmission data is 122,973 KB. Since the complexity of the least squares method far exceeds weighting averages in Experiment 1, the preprocessing data consumed by multiplication and other operations also increase tremendously. Meanwhile, the growth of the execution time maintains an acceptable rate, which is slower than the linear increase of the data size.

4.5.3. Performance comparison

To demonstrate the efficiency of our proposed method, we perform an extensive experiments to compare the performance of our method with existing technologies. We choose the UCI wine quality dataset [33], which is publicly available and widely used in MPC related papers, as the input to compute the linear regression model by MPC. We perform two experiments by our proposed method to compare the performance in the experiment results of [22,25–27], which are also produced with the same wine quality dataset [33]. The experiment results are shown in Table 9. All the configuration information and execution time data in Table 9 are extracted from the evaluation related sections of [22,25–27]. The experiment results of our proposed method with 3 and 10 parties are listed as ①and ⑧, respectively. The experiment results of [22,25,27] are listed as ②, ③, ⑨, respectively. In [26], authors implemented two versions linear regression with Cholesky factorization and Conjugate Gradient Descent (CGD) methods. They evaluated each method with 32-bit and 64-bit fixed-point representations. The experimental results of Cholesky-32-bit, CGD-32-bit, Cholesky-64-bits and CGD-64-bit are listed as ④–⑧, respectively. In Table 9, we present the hardware and software environments of each experiment. The “Split" column presents how data is distributed among data sources. The “Integrity" column indicates if

Table 8
Overall performance metrics of task 3.

Task	Transmission data	Time
Task 3_1	10.17 MB	0.72 s
Task 3_2	6.66 MB	0.42 s
Task 3_3	0.732 MB	1.02 s

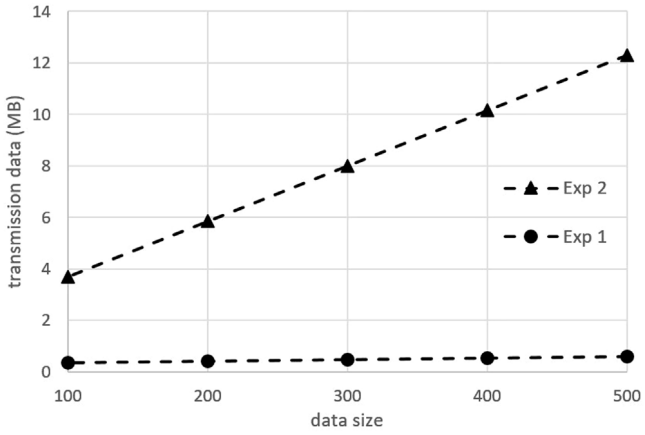

Fig. 2. The transmission data in task 1 and 2.

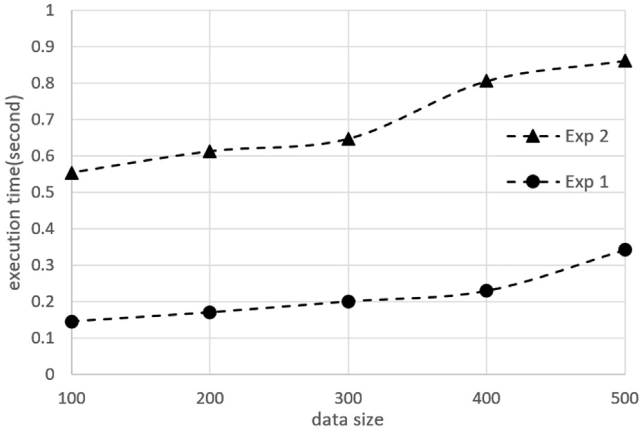

Fig. 3. The execution time of task 1 and 2.

Table 9
Performance comparison.

	HW	SW	Split	Parties	Integrity	Time (s)
①	6 CPU 8G Mem VM	Script SPDZ	row	3	Y	1.98
②	UNK	Java FastGC	UNK	UNK	Y	45
③	16 CPU 224G Mem Azure G4	C++ BOOST GMP	col	2	Y	5.2
④	4 CPU 7.5G Mem EC2 C4	Obliv-C	col	2	N	0
⑤	4 CPU 7.5G Mem EC2 C4	Obliv-C	col	2	N	3
⑥	4 CPU 7.5G Mem EC2 C4	Obliv-C	col	2	N	4
⑦	4 CPU 7.5G Mem EC2 C4	Obliv-C	col	2	N	23
⑧	6 CPU 8G Mem VM	Script SPDZ	row	10	Y	4.05
⑨	40 CPU 500G Mem Server	Python phe gmpy2	row	10	N	4.09

a data source can detect the data tampering of other roles, which is critical for real-world applications. The "Time" column is the execution time of the online computation phase of each method.

From the experimental results in Table 9, we can see that our proposed method is much better than [22] and [25]. Compared with [27] in 10 parties condition, we achieve smaller execution time with much lower hardware configuration. In addition, our proposed method achieves better performance compared with the Cholesky factorization version with 64-bit representation (⑤) and the conjugate gradient descent version with 32-bit (⑥) and 64-bit (⑦) representations of [26]. Although the execution time of the Cholesky factorization version with 32-bit representation (④) in [26] is 0s, it cannot detect data tampering when Crypto Service Provider (CSP) and Evaluator, who are two important roles in that method, are both malicious. This shortcoming makes it not feasible for real-world applications. In contrast, our proposed method can handle dishonest majority case as introduced in 3.2.2. It is important to prevent valuable data from being tampered in real-world applications. In a summary, the experimental results in Table 9 demonstrate the efficiency of our proposed method by comparing to the existing works.

5. Conclusion and future work

Privacy preserving distributed data mining is an important task for big data.. In this paper, we have proposed to utilize the MPC and SPDZ protocol to perform this task. More specifically, we firstly pointed out two mining tasks that are not supported by the current MPC framework, viz., calculating statistics of different types of data and studying the relationship between variables by linear regression. To accomplish these data mining tasks, we have proposed two solutions based on matrix computation with one-hot encoding and LU decomposition. The solutions have been implemented based on the SPDZ secure multi-party computation protocol. To validate the effectiveness and evaluate the performance of our design and implementation, we set up a set of experiments with synthesized data, cotton trading records and vehicle driving logs. The experimental evaluation results have demonstrated that the mining results of our implementation are consistent with those produced by traditional single machine implementation. Meanwhile, the performance evaluation results have demonstrated that our design and implementation are practical for real-world data mining tasks.

Last but not least, it is also worthwhile to mention our work still has potential for improvements. Firstly, the performance of our implementation could be improved by utilizing graphic processing unit (GPU) acceleration. It would be helpful for mining extra large scale dataset. Secondly, we would like to explore more matrix operations besides QR and LU decomposition. It is important to support more complicated data mining tasks. At last, we would like to extend our work from relative simple statistical computation to more machine learning tasks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Jun Liu: Conceptualization, Methodology, Software, Validation, Formal analysis, Resources, Investigation, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Yuan Tian:** Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft. **Yu Zhou:** Validation, Investigation, Writing - original draft. **Yang Xiao:** Data curation, Writing - original draft, Visualization. **Nirwan Ansari:** Writing - original draft, Writing - review & editing, Supervision.

Acknowledgment

This work is supported by the project of User Privacy Protection of Internet of Vehicle Data, No. 18DZ2202500, Science and Technology Commission Shanghai Municipality, China.

References

[1] Y.A.A.S. Aldeen, M. Salleh, M.A. Razzaque, A Comprehensive Review on Privacy Preserving Data Mining, Vol. 4, No. 1, Springerplus, 2015, p. 694.
[2] A. Sachan, D. Roy, P. Arun, An analysis of privacy preservation techniques in data mining, in: Advances in Computing and Information Technology, Springer, 2013, pp. 119–128.
[3] H. Vaghashia, A. Ganatra, A survey: Privacy preservation techniques in data mining, Int. J. Comput. Appl. 119 (4) (2015) 20–26.
[4] J.D. Silva, C. Giannella, Distributed data mining and agents, Eng. Appl. Artif. Intell. 18 (7) (2005) 791–807.
[5] L. Kamm, J. Willemson, Secure floating point arithmetic and private satellite collision analysis, Int. J. Inf. Secur. 14 (6) (2015) 531–548.
[6] M. Aliasgari, M. Blanton, Y. Zhang, A. Steele, Secure computation on floating point numbers, Comput. Arith. (2013) 1–19.
[7] O. Catrina, S.D. Hoogh, Improved primitives for secure multiparty integer computation, in: International Conference on Security & Cryptography for Networks, 2010.
[8] O. Catrina, A. Saxena, Secure computation with fixed-point numbers, in: International Conference on Financial Cryptography and Data Security, Springer, 2010, pp. 35–50.
[9] J. Liu, Y. Liang, N. Ansari, Spark-based large-scale matrix inversion for big data processing, IEEE Access 4 (2016) 2166–2176.
[10] MP-SPDZ, https://github.com/data61/MP-SPDZ.
[11] Y. Zhou, Y. Tian, F. Liu, J. Liu, Privacy preserving distributed data mining based on secure multi-party computation, in: The 11th International Conference on Advanced Infocomm Technology, 2019 in press.
[12] A.C.-C. Yao, Protocols for secure computations, in: FOCS, Vol. 82, 1982, pp. 160–164.
[13] O. Goldreich, Secure multi-party computation, Manuscript. Preliminary version, vol. 78, 1998.
[14] S. Goldwasser, Multi party computations: Past and present, in: Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, PODC '97, 1997, pp. 1–6.
[15] C. Gentry, et al., Fully homomorphic encryption using ideal lattices, in: STOC, Vol. 9, 2009, pp. 169–178.
[16] R. Bendlin, I. Damgård, C. Orlandi, S. Zakarias, Semi-homomorphic encryption and multiparty computation, in: Lecture Notes in Computer Science, vol. 2010 (2010), 2011, pp. 169–188.
[17] I. Damgård, C. Orlandi, Multiparty computation for dishonest majority: From passive to active security at low cost, in: Conference on Advances in Cryptology, 2010.
[18] J.B. Nielsen, P.S. Nordholt, C. Orlandi, S.S. Burra, A new approach to practical active-secure two-party computation, in: Cryptology Conference on Advances in Cryptology-crypto, 2012.
[19] I. Damgård, V. Pastro, N. Smart, S. Zakarias, Multiparty computation from somewhat homomorphic encryption, in: Annual Cryptology Conference, Springer, 2012, pp. 643–662.
[20] W. Du, M.J. Atallah, Secure multi-party computation problems and their applications: A review and open problems, in: New Security Paradigms Workshop, 2001, pp. 13–22.
[21] D. Bogdanov, R. Talviste, J. Willemson, Deploying secure multi-party computation for financial data analysis, in: International Conference on Financial Cryptography and Data Security, Springer, 2012, pp. 57–64.
[22] V. Nikolaenko, et al., Privacy-preserving ridge regression on hundreds of millions of records, in: 2013 IEEE Symposium on Security and Privacy, 2013, pp. 334–348.
[23] R. Hall, S.E. Fienberg, Y. Nardi, Secure multiple linear regression based on homomorphic encryption, J. Off. Stat. 27 (4) (2011) 669.
[24] F.K. Dankar, R. Brien, C. Adams, S. Matwin, Secure multi-party linear regression, in: 2014 EDBT/ICDT Workshops, pp. 406–414.
[25] C. Martine de, et al., Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data, in: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, ACM, 2015.
[26] G. Adria, et al., Privacy-preserving distributed linear regression on high-dimensional data, in: Proceedings on Privacy Enhancing Technologies, 2017, pp. 345–364.
[27] O. Giacomelli, et al., Privacy-preserving ridge regression with only linearly-homomorphic encryption, in: International Conference on Applied Cryptography and Network Security, Springer, Cham, 2018, pp. 243–261.
[28] A. Shamir, How to share a secret, Commun. ACM 22 (1979) 612–613.
[29] N.R. Draper, H. Smith, Applied Regression Analysis, Vol. 326, John Wiley & Sons, 2014.
[30] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, third ed., The MIT Press, 2009.
[32] A. Gil, J. Segura, N.M. Temme, Numerical Methods for Special Functions, Vol. 99, SIAM, 2007.
[33] UCI Machine Learning Repository, Wine Quality Data Set, https://archive.ics.uci.edu/ml/datasets/Wine+Quality.