Hindawi
Security and Communication Networks
Volume 2021, Article ID 6184920, 14 pages
https://doi.org/10.1155/2021/6184920

Research Article

Efficient Noninteractive Outsourcing of Large-Scale QR and LU Factorizations

Lingzan Yu, Yanli Ren, Guorui Feng, and Xinpeng Zhang

School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China

Correspondence should be addressed to Yanli Ren; renyanli@shu.edu.cn

Received 28 April 2021; Accepted 4 June 2021; Published 18 June 2021

Academic Editor: Zhili Zhou

Copyright © 2021 Lingzan Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

QR and LU factorizations are two basic mathematical methods for decomposition and dimensionality reduction of large-scale matrices. However, they are too complicated to be executed for a limited client because of big data. Outsourcing computation allows a client to delegate the tasks to a cloud server with powerful resources and therefore greatly reduces the client's computation cost. However, the previous methods of QR and LU outsourcing factorizations need multiple interactions between the client and cloud server or have low accuracy and efficiency in large-scale matrix applications. In this paper, we propose a noninteractive and efficient outsourcing algorithm of large-scale QR and LU factorizations. The proposed scheme is based on the specific perturbation method including a series of consecutive and sparse matrices, which can be used to protect the original matrix and obtain the results of factorizations. The generation and inversion of sparse matrix has small workloads on the client's side, and the communication cost is also small since the client does not need to interact with the cloud server in the outsourcing algorithms. Moreover, the client can verify the outsourcing result with a probability of approximated to 1. The experimental results manifest that as for the client, the proposed algorithms reduce the computational overhead of direct computation successfully, and it is most efficient compare with the previous ones.

1. Introduction

At the era of big data, the exponential growth of data volume has attracted more and more attention. It is predicted that by 2021, every person will average generate about 1.7 MB of data per second. This presents an inevitable challenge, because the owners of data and analysts need to find a way to store and analyse enormous data properly and effectively.

During the processing of large-scale data, the storage capacity and computing power are closely related to the computer system hardware and available memory. Due to the limitations of the local computation resources, the client can neither be able to employ large-scale data tasks nor meet the requirements for calculation time. Therefore, a potential way to deal with the dilemma is to seek help from advanced computing facilities with great computational power. This promotes the motivation to develop outsourced computing server, also known as cloud server. In an outsourcing computation scheme, the clients only need to upload data and computation tasks to the cloud server, and the server provides on-demand resources. Outsourcing computation allows source-constrained client to the abundant computing resources and save abundant local memory space or computation overhead.

Matrix factorization is a data analysis technique and has been increasing prevalent several fields, such as regression analysis [1] and dimension reduction [2]. QR factorization [3] is one of the basic methods for matrix factorization, and it implements a way of decomposing an original matrix into the product of two specific matrices which satisfy the orthogonality and upper triangular properties, respectively, so that some useful properties of the original matrix can be extracted further. It is common to utilize QR factorization to analyse real-world data analysis problems; for instance, it is always used to employ singular value decomposition [4] and utilized to solve least squares problems [5] as well. Lower-upper (LU) factorization is a means of taking the matrix as the product of the lower triangular matrix and the upper

triangular matrix, it is used wildly in numerical analysis and linear algebra fields, computers usually use LU factorization to solve the square system of linear equations, and it is also an essential step when calculating the inversion of a matrix and the determinant of a matrix.

Notice that QR and LU factorizations have been widely used in many fields, the requirement of conducting QR and LU factorizations on large-scale matrix becomes increasingly common, however, it will cost computational complexity of $O(m^2n)$ when conducting QR or LU factorizations of a $m \times n$ matrix, as the dimension increases, the computation overhead expands sharply, and for resource-constrained clients such as individuals or small companies, the enormous computation and storage overheads are beyond their computing capabilities when dealing with a large-scale matrix. To address this problem, our goal is to design an efficient outsourcing algorithm of large-scale QR and LU factorizations while preserving noninteractive performance, which will help the clients take use of notable advantages of the cloud server to complete complicated task.

In order to achieve our design goals, we need to pay attention to potential menace and several challenges. More precisely, the first is the security issue [6], and the data outsourced to the cloud may be private and sensitive, which means the clients are undesirable to leak their data; our proposed algorithms will take actions based on matrix permutation techniques to transform the original data, to prevent the semi-trustful or even malicious cloud server from deducing the client's data and inferring the behaviour information; besides, our proposed algorithms are designed under one cloud server and avoid the strict security precondition of all the cloud servers must be noncolluding. Another challenge is the correctness of calculations, and the client should check the correctness of results of outsourcing computation, so in our proposed algorithms, we have designed a subalgorithm to determine whether the calculation tasks outsourced to the cloud server are performed correctly or return invalid results simply. In addition, efficiency is also a vital challenge, it is required that the computational overheads locally should be remarkably less than the computational expenses without outsourcing, we utilize a special matrix generation technique with little cost, and only sparse matrix multiplication is required in encryption and decryption algorithms; vector-matrix multiplication is required in the verification algorithm, which lessens the computational burden on the client side successfully. Moreover, the client employs our algorithms without interaction between the client and the cloud server, thereby reducing clients' communication overhead and minimizing the risks of destroying data security and privacy during the interaction process.

In this paper, we put forward a secure and noninteractive outsourcing algorithm of large-scale QR and LU factorizations. Concretely, our algorithm utilizes special sparse matrices to encrypt the original matrix. The client first generates transform matrices for outsourcing QR and LU factorizations at the low computational cost and then encrypts the original matrix to hide the original data through matrix multiplication. After that, the cloud conducts the QR factorization and LU factorization on the transformed matrix without interaction between the clients and sends the computation results back to the client. Once getting the results, the decomposition results will be verified through the Freivalds algorithm by the client. If the results are proved to be correct, the client recovers the desired factor matrices in the way of computing the product of returned results and the inverse of the permutation matrix.

We have summarized all the contributions of our work as follows:

(1) We propose an efficient and noninteractive outsourcing algorithm for large-scale QR and LU factorizations. In terms of complexity, our proposed algorithms realize the reduction of the computational overhead from $O(m^2n)$ to $O(\max\{m^2, mn\})$ on the client side, where m, ndenote the dimension of the matrix. Compared with the previous ones, it is the most efficient outsourcing scheme for the client.

(2) The proposed QR and LU factorization algorithms are also noninteractive which achieves small communication overheads, where several interactions between the client and the cloud server are required in the previous algorithms.

(3) The proposed algorithms use a matrix transform method based on a series of particular and sparse matrices in both encryption and decryption stage, and we prove our method can satisfy the demand of the privacy security for outsourcing.

(4) The verifiability of the results is realized in our proposed scheme, which allows the client to judge whether the results of the outsourcing computation are correct with a probability of close to 1.

The structure of this paper is arranged as follows. Section 3presents the preliminaries including concrete QR and LU algorithms and system architecture. In Section 4, we will introduce the proposed algorithm for secure and noninteractive outsourcing of large-scale QR and LU factorizations in detail. A comprehensive theoretical analysis will be shown in Section 5. We provide the experimental data including the performance of our algorithms and the comparison in Section 6and summarize the algorithms in our paper in Section 7.

2. Related Work

In recent years, driven by the widespread application of the large-scale matrix [7], there exists some outsourcing algorithms for the matrix operations at present, which are designed for complicated matrix computations such as multiplication, matrix inversion, matrix eigenvalue evaluation, and matrix decomposition. These algorithms can be divided into two categories: algorithms based on cryptographic technology and those based on linear transformation. For the former, many scholars propose to use traditional encryption techniques such as homomorphic encryption [8] to protect the client's data privacy. For the latter, scholars proposed data transformation techniques to transform the original data. When it comes to the outsourcing of matrix multiplication calculation, as discussed by Atallah [9], they commenced the research on outsourcing algorithms through

mapping the random permutation matrix to transform the original matrix. Benjamin and Atallah [10] proposed a secure outsourcing scheme of matrix multiplication through employing homomorphic encryption to the original matrix. As for matrix inversion, Mohassel [11] proposed a secure outsourcing algorithm firstly, and due to the constraints of efficiency, their algorithm failed to be practical in applications. A scheme for outsourcing matrix inversion computation based on the novel transformation method is introduced in Lei [12]. Chen et al. [13] further put forward a matrix inversion outsourcing scheme without the precondition that the original matrix must be invertible.

In recent years, the research on matrix factorization outsourcing has attracted considerable attention. An outsourcing protocol for matrix factorization based on the fully and partially homomorphic encryptions is presented in Kim [14]. Duan et al. [15] proposed an outsourcing scheme of computation-intensive nonnegative matrix, which uses multiplication of permuted matrices. As discussed by Fu [16], they used two cloud servers which are noncollusion to outsource nonnegative matrix factorization and take effective encryption measures based on Paillier homomorphism to protect data privacy. Zhou et al. [17] proposed an outsourcing scheme for matrix eigenvalue decomposition and matrix singular value decomposition, which achieved the verifiability of the results. Luo et al. [18] proposed a secure large-scale QR and LU decomposition outsourcing algorithm and protected the original matrix by utilizing specific matrix transformations, which greatly reduces the client's local computing overhead. However, their algorithm requires additional interaction between the client and cloud, which results in high communication overhead. Zhang et al. [19] presented the implementation of QR decomposition based on a ring-LWE homomorphic encryption and reduced the number of interactions between the client and cloud. As we know, the plaintexts to be encrypted can only be integers and the data should round to the nearest integers, and so the accuracy of the scheme cannot be guaranteed. Moreover, the experimental stage of the scheme does not include the performance when the scheme is applied to large-scale matrices. Simultaneously, neither of the above two schemes can provide verification of the outsourced results.

In previous related research, Luo et al. [18] transform the original matrix by multiplying it with two random and dense matrices, one of them is an orthogonal matrix, and when it comes to the orthogonal matrix, it takes $O(m^2)$ computational complexity to generate a random orthogonal matrix and takes $O(m^3)$ computational complexity to get the inversion of this orthogonal matrix, so their scheme chooses to outsource this progress. As for transforming the original matrix, includes dense matrix multiplication, this takes $O(m^2 n)$ complexity [20]. Because dense matrix multiplication and matrix inversion computation require huge resources, they also need to outsource. In a word, they choose to outsource these computations in an interactive way, and every matrix for interaction still needs to be encrypted for enhanced privacy protection, which results in a lot of communication overhead on the client.

3. Backgrounds

In this section, we will provide the details concerning QR and LU factorizations and describe the system model and the threat model.

3.1. Householder-Based QR Factorization. The well-known QR factorization of a matrix will decompose matrix A into the form of the product of an orthogonal matrix and an upper triangular matrix, let A be a real $m \times n$matrix ($m \geq n$), and A may be decomposed into the product on the form

$$A = Q \times R, \tag{1}$$

where Q is the $(m \times n)$ orthogonal ($Q^T Q = I$), and R is the $(n \times n)$ upper triangular; they are obtained by the use of a sequence of Householder transformations, and this type of transformation conducts the QR decomposition using elementary orthogonal Householder matrices, such as

$$P_{m-1} \cdots P_2 P_1 A = R, \tag{2}$$

where P_i is an orthogonal and symmetric matrix called the Householder transformation matrix, and it is obtained by the following formula:

$$P_i = I - 2\frac{v_i v_i^T}{v_i^T v_i}, \tag{3}$$

where v is a column vector and I is the identity matrix of order size (v_i), note that v_1 is the first column vector of original matrix A, we can get P_1 by substituting v_1 into equation (3), and then multiply P_1 by A, and v_2 is the second column vector of new matrix $P_1 A$, we can get P_2 by substituting v_2 into equation (3), v_2 is the third column vector of new matrix $P_2 P_1 A$, following the same process, $v_i$is the ith column vector of the corresponding matrix, P_i can be obtained by substituting v_i into equation (3), and continue this process until it reaches P_m.

And the matrix $Q$is not given explicitly as

$$Q^T = P_{m-1} \cdots P_2 P_1. \tag{4}$$

Since P_i is an orthogonal matrix, we can derive Q as

$$Q = P_1^{-1} P_2^{-1} \cdots P_{m-1}^{-1}. \tag{5}$$

3.2. Gauss Elimination-Based LU Factorization. The well-known LU factorization refers to decompose a matrix Binto two factors, a lower triangular matrix $L$and an upper triangular matrix U. Assume $B$is a real $m \times n$ matrix ($m \geq n$), and it may be decomposed into the product on the form

$$B = L \times U, \tag{6}$$

where L is the $(m \times n)$ lower triangular and U is the $(n \times n)$ upper triangular which means all the elements below the diagonal are zero, and U is obtained with the use of a sequence of Gauss transformations [20]:

$$M_{m-1} \cdots M_2 M_1 B = U, \tag{7}$$

where M_i is obtained by the following formula:

$$M_i = I - \Pi_i e_i. \tag{8}$$

Note that $\Pi_1 = (0, \pi_2^1, \pi_3^1, \ldots, \pi_m^1)^T$, where $\pi_i^1 = (b_{i1}/b11)$, b_{i1} is the ith element of the first column of matrix B, and I is the identity matrix of order size $(\Pi_i e_i)$; let $e_1 = (1, 0, \ldots, 0)$, and we can get M_1 by substituting $\Pi_1$and e_1 into equation (8). After getting a new matrix by multiplying M_1 by B, then we use ith element of the second column of matrix $M_1 B$ to calculate $\Pi_2 = (0, 0, \pi_3^2, \ldots, \pi_m^2)^T$; let $e_2 = (0, 1, \cdots, 0)$, and M_2can be obtained by substituting $\Pi_2$and $e_2$into equation (8). Following the same process, once the last Gauss transformations M_{m-1} are reached. Finally, we can get $U$from equation (7) and the matrix $L$is computed by

$$L = M_1^{-1} M_2^{-1} \cdots M_{m-1}^{-1}, \tag{9}$$

where $M_i^{-1} = I + \Pi_i e_i$.

3.3. Security Requirements

3.3.1. System Model. The architecture of our proposed efficient and noninteractive outsourcing of the large-scale QR and LU factorization algorithm is illustrated in Figure 1, involves two main different entities: the client and the cloud server with excellent computational ability and significant storage capacity. Due to the inability to carry out the matrix factorizations when the size of this matrix is too large, the client would like to outsource this task to the cloud server. The client only needs to complete these processes locally include key generation, encryption, decryption, and verification. As for the cloud server, it will conduct QR and LU factorizations on the encrypted matrix. There is no extra interaction between two entities in our proposed algorithm.

The implementation of our proposed algorithm consists of five subalgorithms. The concrete definitions of these algorithms are shown as follows:

(1) $KeyGen(\lambda) \longrightarrow K$: with the input of a security parameter λ, this algorithm will generate a secret key K which will be further used for matrix encryption and matrix decryption, and the secret key K is stored by the client.

(2) $ClientEnc(A, K) \longrightarrow A'$: with the input of an original matrix $A \in R^{m \times n}$, this algorithm is executed by the client to output the encrypted matrix $A' \in R^{m \times n}$, and the data are encrypted by the secret key K.

(3) $CloudCom(\mathcal{F}, A') \longrightarrow (Q', R')$: receiving the computation task $\mathcal{F}$and the corresponding matrix A', this algorithm is executed by the cloud to complete the task $\mathcal{F}$ and output the result $Q' \in R^{m \times m}$and $R' \in R^{m \times n}$, which are two factor matrices of A'.

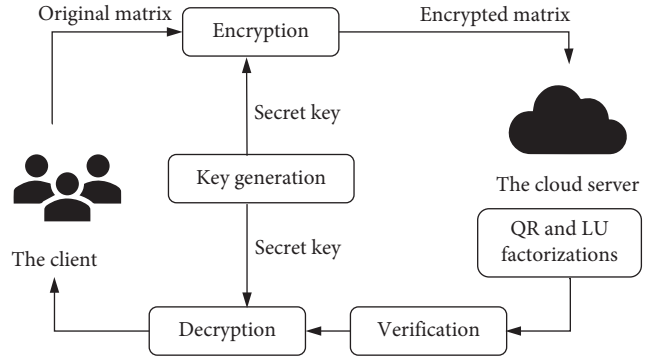

Figure 1: The model of our proposed outsourcing algorithm.

(4) $ClientVer(A', l, Q', R') \longrightarrow (Q', R') \cup \perp$: receiving the result $Q'$and R' from the cloud, this algorithm is executed by the client to verify the correctness of the answer and detect the incorrect answer. If the result passes the verification algorithm, the client accepts the returned results (Q', R'). Otherwise, the algorithm returns $\perp$.

(5) $ClientDecrpyt(K, Q', R') \longrightarrow (Q, R)$: after verifying the result Q' and R', this algorithm is executed by the client, and the client utilizes the secret key K to decrypt the returned result, to recover and output two factor matrices $Q$and $R$for the original matrix A.

3.3.2. Threat Model. As we know, the client can benefit from outsourcing computation, but it still causes challenges and threats during data treatments. We think about the security challenges and threats from two kinds of servers in generic outsourcing frameworks, one is from external attackers, it means the cloud server and the stored data might suffer from external attacks, and this situation is not considered in this paper; in addition, another security threat is from internal participants, it means that the threats to information confidentiality exactly come from the cloud server, and we consider two kinds of threat models which are classified depending on the cloud's behaviour: one is named the "honest-but-curious" [21], the cloud server may be curious and take action to infer confidential information from the original data or the intermediate results, although the cloud server follows the protocol specification and execute the computation faithfully, the second is "malicious" model, and this cloud server may deceive the client and return the client with incorrect answers on purpose in order to save computational resources. And we assume that the cloud is malicious in our proposed scheme.

3.3.3. Design Goals. In this section, we formulate these design goals of our proposed outsourcing algorithm as follows:

(A) Correctness: assume that the client and the cloud server execute the outsourcing scheme faithfully,

and the client can recover the result of the computation and then get the correct result.

(B) Input and output privacy: it is required that the adversary cannot get any knowledge from client's data during the process of carrying out the scheme. In our paper, for making sure the security of outsourcing algorithm, we introduce the formal security definitions of indistinguishability in value under a chosen-plaintext attack (CPA) [22].

All personal information will be considered confidential if for any probabilistic polynomial time adversary $\mathscr{A}$, there exists a negligible function $\mathrm{nelg}(\lambda)$, and the $\mathscr{A}$'s advantage in winning the experiment is

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{Exp}} \leq \mathrm{nelg}(\lambda), \tag{10}$$

where λ is an initial parameter, $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{Exp}} = |\Pr(x = x') - 1/2|$, and the experiment we will proceed further which is explained in Algorithm 1.

4. Efficient Noninteractive Outsourcing of QR and LU Factorizations

As described before, it will waste too much resources in computation and communication aspects in work [18, 19], we should make progress that the client completes the operation for the original matrix with the least computational complexity. Based on these considerations, we decide to use the sparse matrices to execute matrix multiplication operation. It is necessary to protect the confidentiality of the input/output data simultaneously, we take an efficient encryption method on the original data, the client transforms the original matrix by multiplying a series of particular matrices, these appropriate matrices all are sparse enough to ensure efficiency, and ensure data security at the same time; in this way, the client only needs to turn to the cloud one time and receive results, and the interactions are no longer needed. We use these particular matrices due to the specific forms of the factor matrices after decomposing. Because $R$and $L$are both upper triangular matrices, either QR or LU outsourcing factorizations will use a random upper triangular matrix to encrypt data, and the generation of encryption matrix in *KeyGen*will be similar partly. Another encryption matrix will be related to outsourcing algorithms, they are the orthogonal matrix and the lower triangular matrix, respectively.

In this section, we describe noninteractive and efficient outsourcing of large-scale QR and LU factorizations, including the following steps: the client generates a secret key Kwhich comprises stochastic parameters with input security parameter, then the client employs some encryption steps to transform original matrix, the client sends the encrypted matrix which is exactly blind to the cloud server to the cloud, next, the complicated decomposing computation on the encrypted matrix will be completed by the cloud, and the client utilizes the key Kwhich is saved on his own to recover the results if the results after decomposing pass the verification. We will illustrate every step concretely as follows.

4.1. Efficient Noninteractive Outsourcing of Large-Scale QR Factorizations. Given a valid large-scale input matrix $A \in R^{m\times n}$, the client wants to accomplish the QR factorizations with the assistance of the cloud server, and QR factorization will decompose matrix $A \in R^{m\times n}$ into a product of an orthogonal matrix $Q \in R^{m\times m}$ and an upper triangular matrix $R \in R^{m\times n}$, i.e., $A = Q \times R$.

4.1.1. KeyGen. As mentioned before, this algorithm is conducted on the client side, which can produce a secret key $K$and generate an orthogonal matrix $O \in R^{m\times m}$ and an upper triangular matrix $S \in R^{n\times n}$.

Specifically, for perturbing all elements in original matrix $A \in R^{m\times n}$ in an effective way, we take the novel transform based on Givens matrix [23], and the orthogonal matrix $O \in R^{m\times m}$ is composed of kGivens matrices as

$$O = G_1 G_2 G_3 \cdots G_k, \tag{11}$$

where G_i is a Givens matrix ($i = 1, 2, \ldots, k$), and the detailed generation of $O$is shown in Algorithm 2; the client can construct an orthogonal sparse matrix by substituting certain elements located in (i, i), (i, j), (j, i), and (j, j) positions of the identity matrix $I \in R^{m\times m}$ with some certain random parameters $\cos\theta$, $-\sin\theta$, $\sin\theta$, and $\cos\theta$ respectively. And the structure of G_i can be expressed as

$$G = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos\theta & & -\sin\theta & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & \sin\theta & & \cos\theta & & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix}. \tag{12}$$

The detailed generation of O is presented in Algorithm 2 as follows:

Concretely, we have $G \times G^T = I, G^{-1} = G^T$. Since every G_i is an orthogonal and sparse matrix, we can draw the conclusion that the permutation matrix O is an orthogonal matrix certainly, and it is easy to get the inversion of O on the client side, instead of the outsourcing algorithm for matrix inverse in [18].

Next, let us focus on the generation of the upper triangular matrix $S \in R^{n\times n}$. The client can construct a sparse and upper triangular matrix by multiplying a series of random, consecutive upper triangular matrix. S is composed of n elementary matrices as

$$S = C_1 C_2 C_3 \cdots C_n. \tag{13}$$

(1) A client C generates a random matrix A_0 and a key K with the security parameter λ
(2) An adversary $\mathcal{A}$ output (i, j) and two different values a_0, a_1
(3) C picks $x \leftarrow \{0, 1\}$ randomly, replaces the (i, j) element of A_0 with a_x to get A, and calculates the masked task $ClientEnc(A, K)$
(4) $\mathcal{A}$ predicts $x' \leftarrow \mathcal{A}^{ClientEnc(A,\cdot)}(a_0, a_1, i, j, A')$ with the access to the oracle $ClientEnc(A, \cdot)$
(5) Return 1 if $x' = x$ and 0 otherwise
(C) Efficiency: it actually means that the overhead involved in using outsourcing scheme by the client should be substantially less than the overhead required for executing the computational task computation on its own.
(D) Verifiability: it is commonly required that the proposed algorithm can verify correctness of the returned result, the correct results can pass the verification algorithm successfully, and the probability of incorrect results sent back from a malicious cloud server can pass the verification is negligible.

Algorithm 1: A CPA indistinguishability experiment in value.

Input: size m
Output: O
(1) Set O to be an identity matrix
(2) Set $L = \{1, 2, 3, \ldots, m\}$
(3) **for** $k = 1$: $(m/2)$ **do**
(4) set i, j are random integers chosen randomly in L, and remove i, j from L
(5) set θ to be a random real number within $(0, 2\pi)$
(6) set $O_{ii} = O_{jj} = \cos\theta$, $O_{ij} = -\sin\theta$, $O_{ji} = \sin\theta$
(7) **end for**
(8) take the remaining element l in L
(9) set $O_{ll} = 1$

Algorithm 2: Generation of orthogonal perturbation matrix O.

Concretely, the structure of C_i is as follows:

$$C_i = \begin{pmatrix} 1 & & & & \\ & 1 & \cdots & c_i & \\ & & \ddots & \vdots & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}, \quad i = 1, 2, \ldots, n, \tag{14}$$

where C_i is an elementary matrix, and the detailed generation of S is shown in Algorithm 3; we can see that the all nondiagonal elements of $C_i$are zero except c_i, and the row index of the element c_i is bigger than the column index of c_i; note that the element c_i is randomly picked within the interval $(-2^q, 2^q)$. Therefore, c_i is a random variable with its probability density function defined as

$$f_{C_i}(c_i) = \begin{cases} \dfrac{1}{2^{q+1}}, & -2^q < c_i < 2^q, \\ 0, & \end{cases} \tag{15}$$

and the matrices O and S are kept by the client secretly.

4.1.2. ClientEnc. Considering the existence of malicious servers, it is necessary to ensure the confidentiality of the original matrix and the output matrices as well, and we propose to execute a secure and efficient matrix transforming method. A concise way to encrypt $A$is multiplying $A$by O and S on both the left and right sides simultaneously. Then, the client can obtain the final ciphertext matrix $A' \in R^{m\times n}$ as

$$A' = O \times A \times S. \tag{16}$$

In conclusion, the *ClientEnc* algorithm is completed by multiplying a series of consecutive matrices. In virtue of the property of these sparse matrices, the computation overhead required in this algorithm is affordable.

4.1.3. Clou dC om. Once the client gets the ciphertext matrix $A' \in R^{m\times n}$, the client will send it to the cloud. Afterwards, the cloud server computes the QR factorization which is based on Householder reflection; after the computation, the original matrix A' can be decomposed into the product of $Q' \in R^{m\times m}$ and $R' \in R^{m\times n}$, i.e.,

$$A' = Q' \times R', \tag{17}$$

and then the cloud returns Q', R' to the client.

4.1.4. ClientVer. This algorithm is always utilized to verify whether the cloud server follows the specification and executes the computation faithfully. After receiving the results including Q', R', the client needs to determine whether R' is an upper triangular matrix; if Q', R' do not satisfy the correctness of the structure, the algorithm returns $\perp$.

Input: size n
Output: S
(1) Set S to be an identity matrix
(2) **for** $i = 1$: n **do**
(3) set C_i to be an identity matrix
(4) set c_i to be a random real number within $(-2^q, 2^q)$ except 0, and $q > 0$
(5) set $\alpha_i = i$
(6) set β_i is a random integer chosen randomly within $[\alpha_i, n]$
(7) add the element of the α_ith and the β_ith column in matrix C_i to be c_i
(8) set $S = S \times C_i$
(9) end for

Algorithm 3: Generation of elementary perturbation matrix S.

Input: size m
Output: K
(1) Set K to be an identity matrix
(2) **for** $i = 1$: m **do**
(3) set K_i to be an identity matrix
(4) set k_i to be a random real number within $(-2^q, 2^q)$ except 0, and $q > 0$
(5) set $\alpha_i = i$
(6) set β_i is a random integer chosen randomly within $[1, \alpha_i]$
(7) add the element of the α_ith and the β_ith column in matrix K_i to be k_i
(8) set $K = K \times K_i$
(9) **end for**

Algorithm 4: Generation of elementary perturbation matrix K.

With a certain constant l, for $i = 1$ to l, the client generates two column vectors $r_i \in \{0,1\}^n$ randomly and $r_i' \in \{0,1\}^m$ and then computes $e_i = Q' \times (R' \times r_i) - A' \times r_i$ and $e_i' = (Q')^T \times (Q' \times r_i') - I \times r_i'$, in detail; these equals are used to judge whether Q' is an orthogonal matrix and whether $Q' \times R'$ is equal to A'. If two column vectors e_i and e_i' both equal to 0, it means that the result is correct, and the algorithm outputs Q', R'. Else, the algorithm returns $\perp$.

4.1.5. ClientDecrpyt. Once the client affirms that the encrypted QR decomposition result has passed the verification, it will commence this algorithm to decrypt the Q', R' and obtain the results Q, R. Intuitively, the recover algorithm can be accomplished by multiplying the result with the inverse of the corresponding permutation matrix:

$$Q = O^{-1}Q', R = R'S^{-1}. \tag{18}$$

The inversion of Ocan be computed easily due to the properties of orthogonal matrix, and it means $O^{-1} = O^T$. When it comes to the matrix S^{-1}, we can express as $S^{-1} = C_n^{-1} \cdots C_2^{-1} C_1^{-1}$, and these sparse matrices all are elementary matrix, so the inversion of these matrices is very convenient to get, as $C_i^{-1} = \begin{pmatrix} 1 & & & & \\ & 1 & \cdots & -c_i & \\ & & \ddots & \vdots & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$.

4.2. Efficient Noninteractive Outsourcing of Large-Scale LU Factorizations. Similarly, as for the outsourcing algorithm designed for LU factorization, the encryption method for matrix $B \in R^{m \times n}$ is similar to the previous one, through multiplying it with two random matrices which have a specific form on both the left and right sides simultaneously, and we will explain each subalgorithm in the following part.

4.2.1. KeyGen. This algorithm is executed by the client to generate two permutation matrices $K \in R^{m \times m}$ and $V \in R^{n \times n}$. The lower triangular matrix $K \in R^{m \times m}$ can be formed of a series of random elementary matrices as

$$K = K_1 K_2 K_3 \cdots K_m, \tag{19}$$

where the structure of $K_i$is as follows:

$$K_i = C_i = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & \vdots & \ddots & \\ & k_i & \cdots & 1 \\ & & & & 1 \end{pmatrix}, \quad i = 1, 2, \ldots, m. \tag{20}$$

We can know that every K_i is not only a random elementary matrix, but also a lower triangular matrix, the detailed generation of K is demonstrated in Algorithm 4, note that the element k_i is randomly selected from the interval $(-2^q, 2^q)$, and therefore, k_i is also a random variable

and the probability density function of k_i satisfies the following equation:

$$f_{K_i}(k_i) = \begin{cases} \frac{1}{2^{q+1}}, & -2^q < c_i < 2^q. \\ 0, \end{cases} \quad (21)$$

Similarly, we can generate an upper triangular matrix $V \in R^{n\times n}$ as

$$V = V_1 V_2 V_3 \cdots V_n, \quad (22)$$

where every V_i is not only a random elementary matrix which is an upper triangular matrix as well, and the detailed generation of V is demonstrated in Algorithm 3.

4.2.2. ClientEnc. The client permutates the original matrices through multiplying the secret matrices $K \in R^{m\times m}$ and $V \in R^{n\times n}$, and then the client can obtain ciphertext matrix $B' \in R^{m\times n}$ after the conversion as

$$B' = K \times B \times V. \quad (23)$$

4.2.3. CloudCom. Once the client completes the encryption algorithm and gets matrix $B' \in R^{m\times n}$, the client will turn to the cloud. Then, the cloud server will use its powerful resources to accomplish the LU factorization, after the computation, the original matrix B' can transfer into the product of $L' \in R^{m\times m}$ and $U' \in R^{m\times n}$, i.e.,

$$B' = L' \times U', \quad (24)$$

and then the cloud returns L', U' to the client.

4.2.4. ClientVer. After getting the L', U' returned from the cloud, determining whether L' is a lower triangular matrix and whether U' is an upper triangular matrix will be the first step; if L', U' do not satisfy the correctness of the structure, the algorithm returns $\perp$.

Then, input a certain constant l, and for $i = 1$ to l, the client generates a random column vector $r_i \in \{0,1\}^n$ and then computes $e_i = L' \times (U' \times r_i) - B' \times r_i$; in detail, this equal is used to determine whether $L' \times U'$ is equal to B'. If the result of e_i equals to 0, it means that the result is correct, and the algorithm outputs L', U'. Else, the algorithm returns $\perp$.

4.2.5. ClientDecrpyt. Once the client ensures that the encrypted LU decomposition result is correct, it will conduct this algorithm to decrypt the L', U' and obtain the final results L, U. Intuitively, the decryption algorithm can be accomplished by multiplying the result L', U' with the inverse of the corresponding permutation matrix:

$$L = K^{-1}L', U = U'V^{-1}. \quad (25)$$

5. Theoretical Evaluations

In this section, the elaborate theoretical analysis of our proposed algorithm and efficiency comparisons will be shown.

5.1. Security Analysis

5.1.1. Correctness

Theorem 1. *For any valid large-scale input matrix $A \in R^{m\times n}, B \in R^{m\times n}$, our proposed outsourcing algorithms are correct exactly.*

Proof. If the cloud server performs the task correctly, we can get Q' and R' which satisfy $A' = Q' \times R'$, $\mathrm{OAS} = Q' \times R'$, because cloud performs the QR factorization by Householder reflection, and we can know that the result of QR factorization is unique. So, we can get $Q' = O \times Q$ and $R' = R \times S$, when we use equation (18) to decrypt Q' and R', and the results of *ClientDecrpyt* are correct exactly. As for LU outsourcing factorization, we can get L' and U' when the cloud server performs the task correctly, which satisfy $B' = L' \times U'$ and $\mathrm{KBV} = L' \times U'$, because cloud performs the LU factorization by Gauss elimination, and we can know that the result of LU factorization is unique. So, we can obtain $L' = K \times L$ and $U' = U \times V$ through equation (25). Above all, we can say our proposed outsourcing algorithms are correct exactly. □

5.1.2. Privacy Security

Theorem 2. *For any valid large-scale input matrix $A, B \in R^{m\times n}$, the encrypted matrices $A' = O \times A \times S$ and $B' = K \times B \times V$ are computationally indistinguishable under chosen-plaintext attack, and A' or B' is computationally indistinguishable from a random matrix, respectively, which means that our proposed algorithm satisfies the security definition of data privacy.*

Proof. We consider the input privacy firstly, and in other words, it means the probability that an adversary can infer the information of the original matrix A from A' is negligible, where $A' = O \times A \times S$. In the *ClientDecrpyt* algorithm, O and S are permutation matrices, O is an orthogonal matrix based on Givens matrix, and S is an upper triangular matrix specially, where $O = G_1 G_2 G_3 \cdots G_k$ and $S = C_1 C_2 C_3 \cdots C_n$; we can know each G_i processes two rows of A and each S_i processes one column of A. We assume that $A' = O \times A \times S = (O \times A) \times S = A^0 \times S$. The elements in ith row and the kth row of matrix A^0 can be described as follows, which is concealed by arbitrary parameters $\cos\theta$, $\sin\theta$, where $\theta \in (0, \pi)$,

$$\begin{cases} a^0_{ij} = \cos\theta a_{ij} - \sin\theta a_{kj}, \\ a^0_{kj} = \sin\theta a_{ij} + \cos\theta a_{kj}. \end{cases} \quad (26)$$

By the next encryption step, the elements of A' are easy to deduce that

$$a'_{ij} = a^0_{ij} + c_i a^0_{ij'}. \quad (27)$$

As we mentioned before, these parameters $\cos\theta$, $\sin\theta, c_i$ are chosen by the client randomly and all are blind to adversary $\mathscr{A}$, in order to recover a_{ij}, the adversary needs to access $\cos\theta$, $\sin\theta, c_i$ and a_{kj} even if the adversary knows the encryption algorithm *ClientDecrpyt*, however, the number of unknown variables is more than the number of equations, so it is actually impossible to recover a_{ij} from a'_{ij}, in terms of the *ClientDecrpyt* algorithm, the client generates encryption matrices by $O = G_1G_2G_3\cdots G_k$ and $S = C_1C_2C_3\cdots C_n$, due to all θ in $G_i, i \in [1,k]$ satisfy the uniform distribution of $(0,\pi)$, the random parameter c_i in $S$is selected from the interval $(-2^q, 2^q)$ randomly, the total probability that an adversary can recover a_{ij}form $a'_{ij}$is the product of the probability of recovering θ and $c_i$and also need to get the position of another element a_{kj}, so the success probability of an adversary is $1/(\infty\times\infty)$by launching a brute-force attack to obtain the two sets $\{\theta_1,\theta_2,\ldots,\theta_k\}$ and $\{c_1,c_2,\ldots,c_n\}$, and it is small enough that an adversary can recover a_{ij} is definitely impossible.

Now, we consider the input privacy of B'; in other words, we need to prove that the probability that an adversary can infer the information of the original matrix B from $B'$is negligible, where $B' = K\times B\times V$. In the *ClientDecrpyt* algorithm, $K$and $V$are elementary permutation matrices, where $K = K_1K_2K_3\cdots K_m$ and $V = V_1V_2V_3\cdots V_n$; we can know that each K_i processes one row of $B$and each V_i processes one column of B, so we can say that the elementary permutation matrices will process $m$rows and $n$columns of the original matrix in total. We assume that $B' = K\times B\times V = (K\times B)\times V = B^0\times V$. The elements in ith row and the jth row of matrix B^0 can be described as follows, which is concealed by arbitrary parameters k_i, where $k_i \in (-2^q, 2^q)$:

$$b^0_{ij} = k_i b_{i'j} + b_{ij}. \quad (28)$$

By the next encryption step, the elements of $B'$are easy to deduce that

$$b'_{ij} = b^0_{ij} + v_i b^0_{ij'}. \quad (29)$$

Because these parameters k_i, v_i are chosen by the client randomly and all are blind to adversary $\mathscr{A}$, and the random parameters k_i, v_i in K and V are selected from the interval $(-2^q, 2^q)$randomly, the total probability that an adversary can recover b_{ij} form $b'_{ij}$is the product of the probability of recovering k_i and v_i and also need to get the position of another element $b_{i'j}$, so the success probability of an adversary is $1/(\infty\times\infty)$by launching a brute-force attack to obtain the two sets $\{k_1,k_2,\ldots,k_m\}$ and $\{v_1,v_2,\ldots,v_n\}$, and it is small enough that an adversary can recover b_{ij} is definitely impossible.

If there is no oracle *ClientEnc*, we can say that the advantage of adversary $\mathscr{A}$ can be neglected actually. As for a CPA indistinguishability experiment, an adversary $\mathscr{A}$ outputs two different integers a_0, a_1, and $x \leftarrow \{0,1\}$is picked randomly, and *ClientEnc* is calculated and given to adversary $\mathscr{A}$; $\mathscr{A}$ predicts x' with the access to the oracle *ClientEnc*and eventually outputs x', if $x' = x$, it means adversary $\mathscr{A}$ succeeds, and the successful prediction mainly relies on the following cases:

Case1: the oracle chooses the $K$to answer at least one of $\mathscr{A}$'s queries. Because Kwill be randomly chosen for each query, the probability is $\Pr[\text{case 1}] \le \text{negl}(\lambda)$. Thus, adversary $\mathscr{A}$ can tell and get $x' = x$correctly.

Case2: the oracle never chooses the K to answer at least one of $\mathscr{A}$'s queries. As the output after the *ClientEnc* algorithm is random, $\mathscr{A}$ obtains nothing about which of its values was transformed, and it leads to the probability that $x' = x$ is no more than $1/2 + \text{negl}(\lambda)$.

The probability of successful prediction can be deduced as follows:

$$\Pr[\text{Exp} = 1] = \Pr[\text{Exp} = 1|\text{case 1}]\Pr[\text{case 1}] + \Pr[\text{Exp} = 1|\text{case 2}]\Pr[\text{case 2}] \le \Pr[case1] + \Pr[\text{Exp} = 1|\text{case 2}] \le \text{negl}(\lambda) + \frac{1}{2}. \quad (30)$$

As discussed before, the advantage can be neglected actually, it means that this encrypted matrix is computationally indistinguishable from the matrix generated stochastically and the cloud server cannot get any individual private knowledge contained in the original matrix A, B from A', B'. Thus, we can draw the conclusion that the input privacy is well protected. □

Theorem 3. *For any encrypted matrix $A' = O\times A\times S$ and $B' = K\times B\times V$, after the cloud server conducts the QR and LU factorization and gets the factor matrices Q', R' and L', U', the adversary cannot obtain any private information from original matrices Q, R and L, U.*

Proof. Based on the previous analysis, matrices O and S are generated and kept by the client secretly, so the adversary cannot obtain matrices O and S; it is undoubted that O_{-1} and S_{-1} cannot be obtained, so we can say the adversary is unable to recover the factor matrices according to equation (18) without matrices O_{-1} and S_{-1}. As for LU outsourcing factorization, since the adversary cannot obtain the encrypted matrices K and V and cannot obtain K_{-1} and V_{-1} as well, so it is obvious that the adversary is unable to recover L and U according to equation (25) without K^{-1} and V^{-1} can the adversary recover the result according to equation (25). All these lead up to that our proposed algorithm satisfies the security definition of data privacy including input privacy and output privacy. □

5.1.3. Verifiability

Theorem 4. *For any valid large-scale matrix $A \in R^{m\times n}$, our proposed algorithm is verifiable exactly with a probability of approximated to 1.*

Proof. The vectors e_i, e_i' are generated in our concrete method of verification. However, e_i' is used to analyse whether Q is an orthogonal matrix as we need. So, we simply consider the scenario where the equality $e_i = Q' \times (R' \times r_i) - A' \times r_i = \{e_1^{(i)}, e_2^{(i)}, \ldots, e_n^{(i)}\}^T \in \{0,1\}^n$, where $r_i = \{r_1^{(i)}, r_2^{(i)}, \ldots, r_n^{(i)}\}^T \in \{0,1\}^n$. n is the number of columns in the matrix A'. Assume that the answer from the server is wrong, we can get the inequality $Q' \times R' - A' \neq 0$, which further results in $e_i \neq 0$, it means that there will be at least one nonzero element in $\{e_1^{(i)}, e_2^{(i)}, \ldots, e_n^{(i)}\}^T$, and we assume that the nonzero element locates in kth row, $1 \le k \le n$, i.e., $e_k^{(i)} = 0$. We can obtain

$$\Pr\left[e_i = (0,0,\ldots,0)^T\right] \le \Pr\left[e_k^{(i)} = 0\right] \le \frac{1}{2}. \tag{31}$$

Because the above process will be conducted $l$times independently, so

$$\Pr\left[ClientVer\left(A', l, Q', R'\right) \longrightarrow \text{true} | \text{result is incorrect}\right] \le \left(\Pr\left[e_i = (0,0,\ldots,0)^T\right]\right)^l \le \frac{1}{2^l}. \tag{32}$$

The same procedure may be easily adapted to obtain

$$\Pr\left[ClientVer\left(B', l, L', U'\right) \longrightarrow \text{true} | \text{result is incorrect}\right] \le \frac{1}{2^l}. \tag{33}$$

We can draw the conclusion that the incorrect result is impossible to pass the verification, and our proposed algorithm is verifiable exactly with a probability of approximated to 1. □

5.1.4. Efficiency. For any valid large-scale input matrix $A \in R^{m\times n}$, client performs QR and LU factorizations on its own requires $O(2m^2n + m^3)$ and $O((1/3)\ m^2n)$ computational complexity by employing Householder reflection and Gauss elimination methods, respectively. Considering enormous data in practice, the values of m and n will be large extremely, which leads to the high computational complexity which is unacceptable for the client. In our proposed algorithms, the total complexity of the client is much less than $O(2m^2n + m^3)$. Now, we analyse the time cost in our outsourcing algorithms executed by the client thoroughly.

In the *KeyGen* subalgorithm, O and S are permutation matrices generated by client, O is an orthogonal matrix, and S is an upper triangular matrix specially, the detailed generative process is shown in Algorithm 2 and Algorithm 3, and the total cost is $O(m + n)$.

In the *ClientEnc* subalgorithm, client performs the low-complexity matrix transformation to encrypt A with $A' = O \times A \times S$ where $O \in R^{m\times n}, S \in R^{n\times n}$. The matrix multiplication between two dense matrices can be avoided due to O and $S$are two sparse matrices. The complexity of $ClientEnc(A, K) \longrightarrow A'$ is no more than $O(mn)$.

In the *CloudCom* subalgorithm, the cloud completes QR factorization on ciphertext matrix A' which takes computations with $O(2m^2n + m^3)$complexity.

In the *ClientVer* subalgorithm, client needs to calculate vector-matrix multiplication instead of calculate matrix-matrix multiplication, and the verification process needs to be performed at most $l$times. Due to the *ClientVer* subalgorithm only includes vector-matrix multiplication and $l \ll n$, the overall complexity of *ClientVer* on the client side is $O(m^2 + mn)$.

In the *ClientDecrpyt* subalgorithm, we only need the inversion of $O$and $S$to decrypt the results which are received from cloud when the results have passed the *ClientVer* process. And as we talked in Section 4, the inversion matrices of O and $S$are both convenient to generate. Hence, the *ClientDecrpyt* only contains the sparse matrix-matrix multiplication, and we can know that the complexity of decryption is $O(m^2 + mn)$.

The efficiency analysis of LU factorization is similar, and we list the concrete computational complexity of each subalgorithm in the next section. To sum up, our proposed scheme requires computational complexity of $O(\max\{m^2, mn\})$, which is less than the computational complexity of $O(m^2n)$ of conducting QR or LU factorization, so it is fully in line with the design goals of efficiency.

5.2. Efficiency Comparisons. In this section, we further compare our scheme with those of the most related schemes mentioned in Section 2. We demonstrate the concrete analysis in Tables 1 and 2. These tables show the comparison regarding the computation complexity at each step and the whole communication overhead of the proposed outsourcing algorithm for QR and LU factorization, respectively, and other properties as well.

When calculating the complexity of the QR decomposition outsourcing scheme in work [19], which is using homomorphic encryption (HE), the parameter N denotes the polynomial modulus and is always a power of two, such as 1024, 2048, and 4096. And we assume that the computational complexity of modular multiplication is equal to multiplication, but in real scenario, the time complexity of modular multiplication operations is greater than that of multiplication operation.

As we discussed before, we demonstrate the computation complexity in each stage. And we can observe that the computational complexity in each stage in our scheme is smaller than other works, the computational complexity in

Table 1: Comparisons of the outsourcing scheme of QR factorization.

	KeyGen	Client encrypt	Client verify	Client decrypt	Noninteractive	Communication overhead
Work [18]	$O(m^2+m+n)$	$O(3m^2+6mn)$	—	$O(9m^2+4mn)$	No (4 interaction)	$C(5m^2+8mn+n^2)$
Work [19]	$O(N^2)$	$O(mn\times N^2)$	—	$O[(m^2+mn)\times N^2]$	Yes	$C(2m^2+mn)$
Our scheme	$O(m+n)$	$O(m^2+mn)$	$O(m^2+mn)$	$O(m^2+mn)$	Yes	$C(2m^2+mn)$

Table 2: Comparisons of the outsourcing scheme of LU factorization.

	KeyGen	Client encrypt	Client verify	Client decrypt	Noninteractive	Communication overhead
Work [18]	$O(m^2)$	$O(2m^2+2n^2+6mn)$	—	$O(6m^2+2n^2+4mn)$	No (4 interaction)	$C(4m^2+8mn+n^2)$
Our scheme	$O(m+n)$	$O(m^2+mn)$	$O(mn)$	$O(m^2+mn)$	Yes	$C(2m^2+mn)$

the *ClientVer* subalgorithm is affordable as for the client, and the total communication overhead in our scheme only includes the overhead of sending the encryption matrix A' to the cloud server and receiving the results from the cloud. Our proposed scheme realizes the reduction of the computational overhead from $O(m^2n)$ to $O(\max\{m^2, mn\})$ on the client side.

Specifically speaking, compared with the scheme in [18], the client needs to outsource large-scale matrix multiplication and large-scale matrix inversion, more than outsourcing QR and LU factorizations, which inevitably lead to an increase in computation and communication overhead for the client. Meanwhile, their scheme requires many interactions with the cloud, and every matrix for interaction still needs to be encrypted for enhanced privacy protection, which cause the high communication overhead on the client. However, our proposed scheme is noninteractive, so the client only needs to send matrix and receive results, the interactions are no longer needed, and the client can complete outsourcing process locally with little cost by employing our scheme, which not only guarantees the privacy security but also reduces the client's computation and communication overhead.

In [19], they presented the implementation of QR factorizations through a homomorphic encryption method based on ring learning with errors, note that the plaintexts to be encrypted should only be integers, if the inputs are decimals or floating numbers, they first need be enlarged by a scaling factor, rounded to the nearest integers, only in this way, the inputs can accomplish the homomorphic processing, and their algorithm is subject to low efficiency. Furthermore, the experiment in their scheme has not applied to large-scale matrices and fails to be practical in applications, so we cannot evaluate the performance of their scheme. Compared with the existing scheme [19], our scheme does not require additional data operations, and our scheme has more accuracy and efficiency and fulfills extra verification function as well.

To sum up, as for the client, the client has already outsourced the most complex matrix operation to the cloud and enjoys significant cost savings by deploying our proposal, and the detailed experimental results will be presented in the next section. One can see that regarding computational complexity and communication overhead, the proposed scheme reduces the computational overhead of direct computation successfully, and it has considerably better performance than the existing research.

6. Experimental Evaluation

In this section, we provide a concrete experimental analysis on the performance of the proposed algorithms in a real-world scenario. Our experiments are carried out on the computer with 2.7 GHz Intel Core i5-6400 and the memory of 8 GB to simulate the client side, and another computer with an Intel i7-10700 processor running at 2.8 GHz and 16 GB memory is utilized to simulate the cloud server. We implement the proposed algorithms in Matlab.

We implement our proposed algorithms on different data scale by experiment to evaluate the running time for QR and LU factorizations, respectively, and we record and provide the running time in each different situation. Besides, we assume that the time spent for interaction between the client and the cloud can be proximately neglected. Tables 3 and 4 present the processing time overhead for performing QR and LU factorizations, respectively, and we set the size of the original matrix we construct randomly varies from $500 * 300$ to $7500 * 5000$, and we simulate $l = 20$in the subalgorithm *ClientVer*to make a tradeoff between the efficiency and security.

To make our experiment result more concise and intuitive, we pay more attention to the whole treatment time on the client side and the execution time on the cloud server, so we use these parameters. T_{original} denotes the execution time for the client locally employing QR and LU factorizations without the assistant of the cloud. T_{client} is represent for the execution time for the client to apply our proposed algorithms, including the client's time cost of accomplishing the subalgorithms *KeyGen*, *ClientEnc*, *ClientVer* (), *ClientDecrpyt* on its own. T_{cloud} denotes the execution time for the cloud to accomplish QR and LU factorizations. We will take the execution time as the performance metric, and we can describe the performance gain by $T_{\text{original}}/T_{\text{client}}$; this value is utilized to measure the resources saved for the client when employing the outsourcing algorithm; generally, the larger the value is, the more computing resources can be saved. In the experiment result shown in the tables, we will focus on this performance metric.

Table 3: Time and performance comparisons of QR outsourcing factorization.

Dimension $m \times n$	$T_{original}$(s)	T_{client}(s)	T_{cloud}(s)	$T_{original}/T_{client}$
1000 * 1000	44.09	0.25	24.10	176.75
1500 * 1000	109.39	0.40	68.80	273.12
2000 * 1000	191.93	0.51	139.63	373.25
2000 * 2000	519.76	1.04	324.08	498.33
3000 * 2000	1404.68	1.78	901.72	788.29
4000 * 2500	3455.02	3.70	2531.44	934.96
5000 * 3000	9070.54	7.08	5794.17	1281.78
7500 * 5000	46171.21	25.41	32394.06	1816.67

Table 4: Time and performance comparisons of LU outsourcing factorization.

Dimension $m \times n$	$T_{original}$(s)	T_{client}(s)	T_{cloud}(s)	$T_{original}/T_{client}$
1500 * 1000	11.77	0.38	7.21	31.27
1500 * 1500	28.02	0.55	17.94	50.89
2000 * 1500	52.38	0.85	33.45	62.00
2000 * 2000	83.38	1.03	54.45	80.84
2500 * 2500	176.10	1.82	118.07	96.85
3000 * 3000	308.61	2.79	218.82	110.47
4000 * 4000	675.52	6.04	491.67	111.91
5000 * 5000	1495.33	12.01	1005.73	124.48

And the computation time of matrix of different dimension is demonstrated in Figure 2.

Table 4 presents the processing time (in seconds) and performance gain for solving LU factorizations of different scale.

And the processing time of matrix of different dimension is demonstrated in Figure 3.

We can see that the client enjoys significant cost savings by deploying our proposed scheme, and the required time of the proposed outsourcing scheme increases as the size of the input matrices increase. As can be seen from two tables above, $T_{original}/T_{client}$ which is represented for the resource advantage of the client is up to 1816.67, and it can be considered as our outsourcing scheme has significant computing savings compared to the situation performing QR and LU factorizations on its own.

To validate the efficiency of our scheme sufficiently, we simulate the outsourcing schemes in prior work on our computer and compare the execution time T_{client} with respect to dimension of input matrix in Tables 5 and 6.

As for the outsourcing of QR factorizations, the scheme in work [18] requires interactions with the cloud, and every matrix for interaction still need to be encrypted for enhanced privacy protection; it causes a lot of computational overheads on the client side undoubtedly, and the operations for the elements of matrix in work [19] are so complicated that the execution time T_{client} is the longest among all the works. We can see that T_{client} of our scheme is significantly less than others even though our scheme requires additional overhead of verification, compared to work [18] and work [19], and we can say that our scheme is highly efficient.

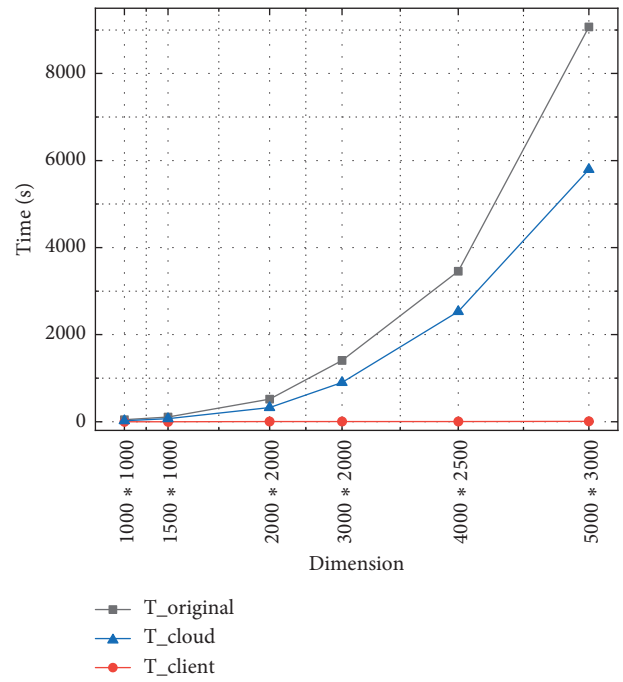

Figure 2: Time comparisons of QR outsourcing factorizations in the proposed algorithm.

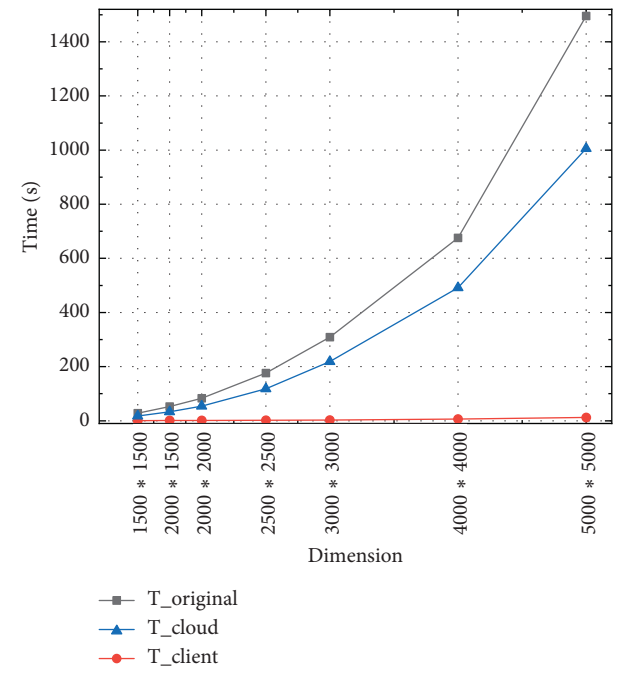

Figure 3: Time comparisons of LU outsourcing factorizations in the proposed algorithm.

The data in Tables 5 and 6 validate our previous analysis. The execution time T_{client} in our scheme is significantly less than others. To sum up, our scheme is the most efficient among the prior work.

According to the tables and figures above, we can conclude that our proposed scheme achieves considerable savings in terms of computation and communication

Table 5: The comparisons of T_{client}in different QR outsourcing schemes.

Dimension $m \times n$	Our scheme (s)	Work [18] (s)	Work [19] (s)
1000 ∗ 1000	0.25	1.24	2.06
1500 ∗ 1000	0.40	2.01	3.93
2000 ∗ 1000	0.51	3.82	6.37
2000 ∗ 2000	1.04	4.29	8.44
3000 ∗ 2000	1.78	12.68	20.16
4000 ∗ 2500	3.70	26.76	40.58
5000 ∗ 3000	7.08	50.11	72.02
7500 ∗ 5000	25.41	155.72	210.68

Table 6: The comparisons of T_{client}in different LU outsourcing schemes.

Dimension $m \times n$(s)	Our scheme (s)	Work [18]
1500 ∗ 1000	0.38	1.29
1500 ∗ 1500	0.55	1.97
2000 ∗ 1500	0.84	3.07
2000 ∗ 2000	1.03	4.38
2500 ∗ 2500	1.82	7.54
3000 ∗ 3000	2.79	12.12
4000 ∗ 4000	6.04	27.84
5000 ∗ 5000	12.01	56.86

overhead on the client side, and the client resource advantage, i.e., $T_{original}/T_{client}$, improves generally related to the increasing size of the matrix as illustrated in tables and figures above, which fulfills totally the design goals for outsourcing large-scale matrix. More importantly, the experiment shows that the proposed scheme needs fewer computing resources than the prior works, and the client employs our algorithm without interaction, thereby reducing clients' communication overhead and minimizing the risks of destroying data security and privacy during the interaction process. In summary, the experiment validates the efficiency of proposed outsourcing scheme by numerical experiments.

7. Conclusions

As the emergence of the big data and cloud computing era, secure outsourcing services are inevitably becoming prevalent, which allows a resource-constrained client moves the heavy computation to the cloud server which has outstanding performance. Considering that QR and LU factorizations have made noticeable contributions in many fields, but they take too much burden and overhead for the client when the dimension of the input matrix is large extremely. To this end, we propose an efficient and noninteractive outsourcing scheme of large-scale QR and LU factorizations. Specially, we take an efficient encryption method on the original data that the client transforms the original matrix by multiplying a series of particular matrices, these matrices all are sparse enough to ensure efficiency and ensure data security at the same time, this techniques can not only lessen the computational overhead of encryption and decryption stage but also deal with serious potential security menace, then the cloud performs matrix factorizations on the transformed matrix, the client just needs to send matrix and receive results, and the interactions are no longer needed, which brings benefit of reducing communication overhead. Our proposed scheme can fulfill the design goals of correctness, privacy preserving, efficiency, and verifiability, and we provide a concrete experimental result on the performance on large-scale matrices. The evaluation results show that regarding computational complexity and communication overhead, our proposed scheme has considerably better performance than the existing research studies.

Data Availability

No data were used to support the findings of this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work described in this paper was supported by the National Natural Science Foundation of China (grant no. 62072295) and Natural Science Foundation of Shanghai (20ZR1419700).

References

[1] C. Gatu and E. J. Kontoghiorghes, "Parallel algorithms for computing all possible subset regression models using the qr decomposition," *Parallel Computing*, vol. 29, no. 4, pp. 505–521, 2003.

[2] J. Jieping Ye, Q. Qi Li, H. Hui Xiong, H. Park, R. Janardan, and V. Kumar, "IDR/QR: an incremental dimension reduction algorithm via QR decomposition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 9, pp. 1208–1222, 2005.

[3] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, USA, 4th edition, 2013.

[4] R. Mathias and G. W. Stewart, "A block QR algorithm and the singular value decomposition," *Linear Algebra and Its Applications*, vol. 182, pp. 91–100, 1993.

[5] A. R. Benson, D. F. Gleich, and J. Demmel, "Direct QR factorizations for tall-and-skinny matrices in MapReduce architectures," in *Proceedings of the 2013 IEEE International Conference on Big Data*, pp. 264–272, Silicon Valley, CA, USA, October 2013.

[6] D. Wang, N. Wang, P. Wang, and S. qing, "Preserving privacy for free: efficient and provably secure two-factor authentication scheme with user anonymity," *Information Sciences*, vol. 321, pp. 162–178, 2015.

[7] Y. Yang, X. Huang, X. Liu et al., "A comprehensive survey on secure outsourced computation and its applications," *IEEE Access*, vol. 7, pp. 159426–159465, 2019.

[8] J. R. Troncoso-Pastoriza and F. Perez-Gonzalez, "Secure signal processing in the cloud: enabling technologies for privacy-preserving multimedia cloud processing," *IEEE Signal Processing Magazine*, vol. 30, no. 2, pp. 29–41, 2013.

[9] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. E. Spafford, "Secure outsourcing of scientific computations," *Advances in Computers*, vol. 54, pp. 215–272, 2002.

[10] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*, pp. 240–245, Fredericton, Canada, October 2008.

[11] P. Mohassel, "Efficient and secure delegation of linear algebra," *IACR Cryptology ePrint Archive*, p. 605, 2011.

[12] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to A public cloud," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, p. 1, 2013.

[13] Z. Chen, A. Fu, K. Xiao, M. Su, Y. Yu, and Y. Wang, "Secure and verifiable outsourcing of large-scale matrix inversion without precondition in cloud computing," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, Kansas City, MO, USA, May 2018.

[14] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, "Efficient privacy preserving matrix factorization via fully homomorphic encryption," in *Proceedings of the ACM AsiaCCS'16*, pp. 617–628, Xi'an, China,, May-June 2016.

[15] J. Duan, J. Zhou, and Y. Li, "Secure and verifiable outsourcing of nonnegative matrix factorization (NMF)," in *Proceedings of the ACM IH & MMSEC*, pp. 63–68, Vigo, Spain, June 2016.

[16] A. Fu, Z. Chen, Y. Mu, W. Susilo, Y. Sun, and J. Wu, "Cloud-based outsourcing for enabling privacy-preserving large-scale non-negative matrix factorization," *IEEE Transactions on Services Computing*, vol. 99, p. 1, 2019.

[17] L. Zhou and C. Li, "Outsourcing eigen-decomposition and singular value decomposition of large matrix to a public cloud," *IEEE Access*, vol. 4, pp. 869–879, 2017.

[18] C. Luo, K. Zhang, S. Salinas, and P. Li, "SecFact: secure large-scale QR and LU factorizations," *IEEE Transactions on Big Data*, p. 1, 2017.

[19] Y. Zhang, P. Zheng, and W. Luo, "Privacy-Preserving outsourcing computation of QR decomposition in the encrypted domain," in *Proceedings of the 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 389–396, Rotorua, New Zealand, August 2019.

[20] A. George and E. Ng, "On the complexity of sparse QR and LU factorization of finite-element matrices," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 5, pp. 849–861, 1988.

[21] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game,," in *Proceedings of the STOC ACM*, New York, NY, USA, January 1987.

[22] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, CRC Press, Boca Raton, FL, USA, 2015.

[23] S. Zhang, S. Pan, W. Wang, and Q. Wang, "A fast secure outsourcing of ridge regression based on singular-value decomposition," in *Proceedings of the 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, Orlando, FL, USA, November 2018.