Hindawi
Security and Communication Networks
Volume 2021, Article ID 6632518, 12 pages
https://doi.org/10.1155/2021/6632518

Research Article

Publicly Verifiable Outsourcing Computation for QR Decomposition Based on Blockchain

Huimin Wang,[1] **Dong Zheng**,[1,2] **and Qinglan Zhao**[1]

[1]*National Engineering Laboratory for Wireless Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China*
[2]*Westone Cryptologic Research Center, Beijing 100070, China*

Correspondence should be addressed to Qinglan Zhao; zhaoqinglan@foxmail.com

Received 9 December 2020; Revised 7 February 2021; Accepted 8 March 2021; Published 24 March 2021

Academic Editor: Jianting Ning

In the Big Data Era, outsourcing computation has become increasingly significant as it supplies computation resources for clients with limited resources. However, there are still many security challenges such as payment fairness, privacy protection, and verification. In this paper, we propose a secure publicly verifiable outsourcing computation scheme for the large-scale matrix QR decomposition. In the proposed scheme, client can pay for outsourcing services through blockchain-based payment system which achieves the payment fairness. Moreover, to protect privacy, both permutation matrix and block diagonal matrix are applied in encryption process. Meanwhile, to achieve the public verification, the computational complexity is reduced by using the matrix digest technology. It is worth mentioning that our scheme is provable and secure under the co-CDH assumption.

1. Introduction

Cloud computing, a new computing technology and service concept, has appeared in the public's vision and serves customers in a pay-per-use manner [1–3]. It has promoted the development of the emerging fields such as smart medical systems in recent years.

Outsourcing computation, as one of the basic applications of cloud computing, can reduce significantly the clients' computational burden [4]. There are two parts, including payment and computation, in outsourcing scheme. For payment part, it often requires online payment and relies on trusted third party such as bank. To realize secure and fair payment of outsourcing services without relying on any third party, fair payment framework based on blockchain has been used for outsourcing services in cloud computing [5]. For computation part, service requester submits the data-to-service provider, which might get service requester's privacy from the data. Therefore, there exist many security challenges during the outsourcing process.

About the protection of client privacy, computing tasks authorized to cloud server involve some important sensitive information frequently, such as core technology of a company and patient health records. So, it is important for users to conceal their data information before uploaded to the cloud server. The previous works have attempted to protect the confidentiality of the data. For example, full homomorphic encryption [6], a cryptographic technique, can allow service provider to perform valid and meaningful operations on ciphertext. However, the existing schemes based on FHE suffer from high computation complexity.

Moreover, the result verification is vital as well. Since the process of cloud computing is not transparent to the public who will upload their data, the public should detect in time whether there are any errors in outsourcing result. There may be many reasons to produce an invalid and wrong result, such as hardware malfunction, software bugs, or malicious hackers. Furthermore, a semihonest cloud server [7] might work dishonestly or even cut down calculation steps due to huge benefits.

Considering financial expenses, an outsourcing computating scheme should be highly efficient. That is, the user's computation in the outsourced process is far less than the computation of their task directly. Otherwise, the outsourcing will get meaningless for the client.

Matrix computation has many applications in scientific and engineering fields. The outsourcing matrix computation also involves the above security challenges. We research on secure outsourcing matrix QR decomposition computation and propose a publicly verifiable scheme based on blockchain. The system consists of two parts: blockchain-based payment and publicly verifiable computation. In this paper, we focus on designing publicly verifiable computation scheme and we refer the reader to reference [5] to know more about blockchain-based payment.

1.1. Contributions. The contribution of this paper can be described from the following three points:

(i) We multiply a sparse block diagonal matrix with the original matrix to protect the client's privacy. The computational complexity is $O(n^2)$ in the encryption process.

(ii) The scheme provides public verification. To reduce the workload of the verifier, we use matrix digest technique which transforms any matrix into a specific vector with chosen parameter in the verification of QR decomposition.

(iii) We show the soundness of the scheme through detailed theoretical analysis, including correctness, security, and efficiency. It is proved that the scheme achieves secure under *co*-CDH difficulty assumption.

1.2. Related Work. Looking back on the development of outsourcing computation in the past decades, many schemes have been designed for different scientific computations. Atallah et al. [8] proposed the concept of the scientific computing outsourcing firstly. To protect privacy of clients, researchers have devoted to design the secure outsourcing schemes [9–14]. Salinas et al. [9] mentioned a privacy-preserving transformation method by adding random matrix to original matrix. For the verifiability of the outsourcing results, Golle and Mironov [15] firstly realized this goal in their scheme. Then, a verifiable scheme about any random function was designed by Gentry [6] which provided the formal concept of verifiable computing. Banabbas et al. [16] put forward to a verifiable scheme for high-degree polynomial functions.

Nevertheless, in many applications, verification needs to be public. In other words, any customer can verify it. Recently, some experts turned their attention to public verifiable computation. Fioreand Gennaro [17] allowed service requester to verify the result with a noninteractive evidence. Meanwhile, Parno et al. [18] gave the concept of the correctness and security which had established a connection between public verification computation and attribute-based encryption (ABE). In addition, Fiore and Gennaro [17] also designed the matrix multiplication outsourcing scheme according to Yao's Garbled Circuits [19]. Different from traditional scheme [11], the scheme [20] has achieved that all clients can share a common matrix M to perform matrix multiplication, which did not protect the security of the original matrix.

Jia et al. [12] took the privacy protection into account, where the matrix can be arbitrary. Li et al. [21] improved its efficiency compared to previous work and achieved the public verification. Zhang et al. [13] reduced the computing overhead in the verification process hugely. The scheme in [22] not only achieved the public verification but also protected privacy information of original data, where matrix digest was utilized to reduce the overhead of key generation and cloud computing.

The above results [12, 17, 20–22] are about publicly verifiable solutions for matrix multiplication. However, there is no publicly verifiable solution about matrix decomposition. Matrix decomposition, as one of the basic matrix operations, has many application scenarios [23–26]. Luo et al. [27] had designed a secure algorithm for QR decompositions without the public verification achieved. We propose a scheme which achieves the promising public verification under the amortized model for QR decomposition of large-scale matrices. To protect privacy, sparse matrices which cut down the computational complexity from $O(n^3)$ to $O(n^2)$ are applied during encryption.

1.3. Organization. In Section 2, it introduces related definitions of verifiable computing and significant mathematical knowledge. Section 3 details the proposed scheme for the publicly verifiable computation of the QR decomposition. The correctness, security, and efficiency analysis are shown in Section 4. At last, it ends up with our conclusion in Section 5.

2. Preliminaries

In this part, we introduce some related definitions, mathematical knowledge, and important techniques.

2.1. Publicly Verifiable Computation. As mentioned by Gennaro [28], a public verifiable computation scheme $\mathscr{VC}$ not only allows a client to outsource his computating task but also states that the outsourcing result is correct and verifiable. The formal definitions of these properties for public verifiable computation are presented in [22, 28]. For the sake of integrity, we give some related definitions before introducing our scheme.

Definition 1. A outsourcing scheme $\mathscr{VC}$ consists of the following five subalgorithms:

(i) $\mathbf{KeyGen}(1^\lambda, \mathscr{F}) \longrightarrow$ (SK, PK):

Given the random selected parameter λ, a public key PK is produced to protect the function $\mathscr{F}$. Simultaneously, a private key SK saved by the service requester secretly is generated by this algorithm.

(ii) $\mathbf{ProbGen}_{\mathrm{SK}}(x) \longrightarrow (\tau_x, \sigma_x)$:

The client performs the encryption together with the SK and gets a decoding value τ_x stored

confidentially, where the input x of function is encoded into a encrypted result σ_x.

(iii) $\mathbf{Compute}_{\mathrm{PK}}(\sigma_x) \longrightarrow (\sigma_y)$:
According to the PK and the encrypted σ_x, the outsourcing server provider produces a blinded output σ_y.

(iv) $\mathbf{Verify}_{\mathrm{SK}}(\tau_x, \sigma_y) \longrightarrow (y \cup \perp)$:
Based on σ_y and τ_x, if σ_y of function $\mathcal{F}$ is correct, it outputs y. Otherwise, it outputs the symbol $\perp$.

(v) $\mathbf{Solve}_{\mathrm{SK}}(\tau_x, \sigma_y) \longrightarrow (y)$:
The algorithm decodes σ_y to generate the final result $y = \mathcal{F}(x)$ with SK and τ_x.

Next, we focus on these properties in publicly verifiable computation scheme $\mathcal{VC}$, including correctness, security, privacy, and efficiency.

Definition 2 (correctness). For a function $\mathcal{F}$, we say the verifiable outsourcing scheme $\mathcal{VC}$ is correct if the key generation algorithm generates keys $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathbf{KeyGen}(\mathcal{F}, 1^{\lambda})$ and satisfies the following condition: $\forall x \in \mathrm{Domain}(\mathcal{F})$, $y = \mathcal{F}(x) \leftarrow \mathbf{Verify}_{\mathrm{SK}}(\tau_x, \sigma_y)$ if $(\sigma_x, \tau_x) \leftarrow \mathbf{ProbGen}_{\mathrm{SK}}(x)$ and $\sigma_y \leftarrow \mathbf{Compute}_{PK}$.

The formalized definition of security of a verifiable computation outsourcing scheme $\mathcal{VC}$ is introduced, where a malicious server cannot persuade the verifier to output a invalid result $\hat{y}$ according to the function $\mathcal{F}$ and input x, e.g., $\mathcal{F}(x) \neq \hat{y}$.

Now, we abstract this objective fact with an experiment which is expressed as below.

Experience $\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Verify}}[\mathcal{VC}, \mathcal{F}, \lambda]$;
For $i = 1, \ldots, l = \mathrm{poly}(\lambda)$;
$(\mathrm{PK}, \mathrm{SK})^{R} \leftarrow \mathbf{KeyGen}(\mathcal{F}, \lambda)$;
$x_i \leftarrow \mathcal{A}(\mathrm{PK}, x_1, \delta_1, \ldots, x_{(i-1)}, \delta_{(i-1)})$;
$(\delta_i, \tau_i) \leftarrow \mathbf{ProbGen}_{\mathrm{SK}}(x_i)$
$(i, \hat{\delta}_y) \leftarrow \mathcal{A}(\mathrm{PK}, x_1, \delta_1, \ldots, x_l, \delta_l)$
$\hat{y} \leftarrow \mathbf{Verify}_{\mathrm{SK}}(\tau_i, \hat{\delta}_y)$;
If $\hat{y} = \perp$ and $\hat{y} \neq \mathcal{F}(x_i)$, output 1, else 0.

Here, $\mathrm{poly}(\cdot)$ is defined as a polynomial.

Given an oracle access, the adversary can produce the encryption of multiple problems. Considering a known input, the adversary can persuade the verifier to work smoothly, where any error is unable to be detected in the output.

Definition 3 (security). For a verifiable computation outsourcing scheme $\mathcal{VC}$, the capability of an adversary $\mathcal{A}$ in the above experiment is defined as follows:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Verify}}(\mathcal{VC}, \mathcal{F}, \lambda) = \Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Verify}}(\mathcal{VC}, \mathcal{F}, \lambda) = 1\right]. \tag{1}$$

For a function $\mathcal{F}$, the scheme $\mathcal{VC}$ is secure if, for any adversary $\mathcal{A}$ running in probabilistic polynomial time (PPT),

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Verify}}(\mathcal{VC}, \mathcal{F}, \lambda) \leq \mathrm{negli}(\lambda), \tag{2}$$

where $\mathrm{negli}(\cdot)$ is a negligible function of its input.

If the outputs of the ProbGen algorithm over two different inputs are indistinguishable, we think a $\mathcal{VC}$ scheme is private. To define privacy of a verifiable computation scheme, we need an experiment. Given the public key PK for the scheme, the adversary $\mathcal{A}$ treats x_0 and x_1 as two inputs randomly. Then, he is given the encoded version of one of x_0 and x_1 and must guess which one was encoded. In the process, the oracle $\mathrm{PubProbGen}_{\mathrm{SK}}(x)$ calls $\mathrm{ProbGen}_{\mathrm{SK}}(x)$ to obtain (δ_x, τ_x) and returns only the public part δ_x. Now, the experiment is described below.

Experience $\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Priv}}[\mathcal{VC}, \mathcal{F}, \lambda]$;
$(\mathrm{PK}, \mathrm{SK}) \overset{R}{\leftarrow} \mathbf{KeyGen}(\mathcal{F}, \lambda)$;
$(x_0, x_1) \leftarrow \mathcal{A}^{\mathrm{PubProbGen}_{\mathrm{SK}}(\times)}(\mathrm{PK})$
$(\delta_0, \tau_0) \leftarrow \mathbf{ProbGen}_{\mathrm{SK}}(x_0)$
$(\delta_1, \tau_1) \leftarrow \mathbf{ProbGen}_{\mathrm{SK}}(x_1)$
$y \overset{R}{\leftarrow} \{0, 1\}$;
$\hat{y} \leftarrow \mathcal{A}^{\mathrm{PubProbGen}_{\mathrm{SK}}(\times)}(\mathrm{PK}, x_0, x_1, \delta_y)$
If $y = \hat{y}$, output 1, else 0.

Definition 4 (privacy). According to the above experiment, the ability of an adversary $\mathcal{A}$ is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Priv}}(\mathcal{VC}, \mathcal{F}, \lambda) = \left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{Priv}}(\mathcal{VC}, \mathcal{F}, \lambda) = 1\right] - \frac{1}{2}\right|. \tag{3}$$

A verifiable computation scheme is private if, for a function $\mathcal{F}$ any adversary $\mathcal{A}$ running in PPT,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Priv}}(\mathcal{VC}, \mathcal{F}, \lambda) \leq \mathrm{negli}(\lambda), \tag{4}$$

where $\mathrm{negli}(\cdot)$ is a negligible function of its input.

Definition 5. (efficiency). A verifiable scheme $\mathcal{VC}$ must be highly efficiency for the client. That is, the time for encryption and verification in the scheme should be shorter than the time to accomplish the computing task directly by itself.

2.2. Bilinear Pairings. The knowledge about bilinear pairings, in a verifiable computating scheme $\mathcal{VC}$, will be introduced as follows.

Let G_1, G_2, and G_T be three multiplicative cyclic groups with the same large prime order p and g_1 and g_2 be generators of G_1 and G_2, respectively. A bilinear pairing is a map $e: G_1 \times G_2 \longrightarrow G_T$, which has the following three characteristics:

(i) Bilinearity: for any $\alpha, \beta \in Z_p$ and $g^{\alpha} \in G_1, h^{\beta} \in G_2$, the equation $e(g^{\alpha}, h^{\beta}) = e(g, h)^{\alpha\beta}$ holds

(ii) Computability: there exist a valid algorithm for solving $e(g,h)$ for any $(g,h) \in G_1 \times G_2$

(iii) Nondegeneracy: for any $g \in G_1$, if, for all $h \in G_2$, equation $e(g,h) = 1$ is true, $g = 1$, where g and h can be interchanged

According to the above, the related definitions about computational assumptions can be described as follows.

Definition 6 (co-CDH problem). Given $g, g^{\alpha} \in G_1, h, h^{\beta} \in G_2$, and $\alpha, \beta \in_R F_p^*$, compute $g^{\alpha\beta}$.

Definition 7 (co-CDH assumption). Given $g, g^{\alpha} \in G_1$ and $h, h^{\beta} \in G_2$, for randomly selecting $\alpha, \beta \in_R F_p^*$, if the probability to compute $g^{\alpha\beta}$ is negligible in any PPT, the co-computational Diffie–Hellman assumption holds in G_1.

2.3. Matrix Digest Technique. As an one-way irreversible mapping process, matrix digest [22, 29] refers to transform an any matrix into a specific vector with a chosen parameter, which makes computational complexity reduce from $O(n^3)$ to $O(n^2)$. In fact, a matrix consists of some column vectors, in which a vector is also a special matrix. For example, a square matrix $\widehat{A}$ can be denoted as $\widehat{A} = (\overrightarrow{a}_1, \ldots, \overrightarrow{a}_i, \ldots, \overrightarrow{a}_n)$, where $\overrightarrow{a}_i \in Z^{n\times 1}$ is a column vector. By this novel technique, we can transform the matrix $\widehat{A}$ into the vector $\overrightarrow{b}$ by a row vector $\overrightarrow{s} \in Z^*$, where the vector $\overrightarrow{b} = \overrightarrow{s}\widehat{A}$ is a matrix digest of $\widehat{A}$.

There are three properties of matrix digest:

(i) Deterministic: the matrix digest of a matrix will be determined uniquely by the known parameter vector, *i.*e., if $\overrightarrow{s}$ is chosen as the parameter, $\overrightarrow{b}$ must be unique for $\widehat{A}$.

(ii) Computable: the result of matrix digest essentially is a vector and retains the computing ability of the initial matrix.

(iii) Irreversible: given a matrix digest, it is difficult for anyone to detect initial matrix and selected parameter. Furthermore, if the matrix digest and the parameter are known at the same time, the initial matrix cannot be obtained as well.

3. The Proposed Scheme

3.1. Treat Model. The semihonest model introduced in [7] is an honest but curious one with an untrustworthy cloud server as the main adversary. It was also mentioned in [30], where participants in the outsourcing are required to honestly execute the designed scheme. With returning a correct result, semihonest cloud will try to recover sensitive information of the data. Our scheme is based on a semihonest cloud server and introduces an independent data center which is trustworthy.

3.2. System Model. Considering the public verification, we give a system model about outsourcing computation with the following five entities introduced, as shown in Figure 1.

(i) Data center ($\mathscr{DC}$): some keys are produced by $\mathscr{DC}$. After initializing parameters, it generates the private keys and some public keys. Next, it takes advantage of the private key to generate the evaluation key for $\mathscr{CS}$. Finally, it sends the private key to $\mathscr{C}$ and $\mathscr{V}$ over the secure channel.

(ii) Client ($\mathscr{C}$): first of all, $\mathscr{C}$ should deposit enough money into $\mathscr{B}_{\mathscr{P}}$ for the cost of outsourcing services. Meanwhile, a request is sent to $\mathscr{CS}$ about solving a QR decomposition of the large-scale matrix. To protect privacy, $\mathscr{C}$ needs to encode the original matrix before the private matrix is uploaded to $\mathscr{CS}$. Then, a verification key should be generated for $\mathscr{V}$.

(iii) Cloud server ($\mathscr{CS}$): like $\mathscr{C}$, $\mathscr{CS}$ also needs to provide deposits to $\mathscr{B}_{\mathscr{P}}$. As service provider, $\mathscr{CS}$ needs to perform QR decomposition of the encryption matrix and earn fees from $\mathscr{C}$. Moreover, a proof sent to $\mathscr{V}$ together with computating results is generated by using the evaluation key. Finally, the result matrices will be transmitted to $\mathscr{C}$. If $\mathscr{C}$ has no objection to the outsourcing result within a specified time, $\mathscr{CS}$ will provide a proof $\mathrm{OS}_{\mathrm{end}}$ to $\mathscr{B}_{\mathscr{P}}$ and get the corresponding fees. Otherwise, $\mathscr{CS}$ provides compensation to $\mathscr{C}$.

(iv) Verifier ($\mathscr{V}$): any verifier can be regarded as $\mathscr{V}$. Utilizing the verification key and the proof, $\mathscr{V}$ will examine the correctness of the outsourcing results.

(v) Blockchain payment ($\mathscr{B}_{\mathscr{P}}$): we take advantage of the payment system based on blockchain $\mathscr{B}_{\mathscr{P}}$. After receiving deposit from $\mathscr{C}$ and $\mathscr{CS}$, $\mathscr{B}_{\mathscr{P}}$ provides a proof OS_{start} for $\mathscr{CS}$ to confirm to start the outsourcing service.

3.3. Process Description. The system model consists of two parts: blockchain-based payment system and publicly verifiable outsourcing computing system. In blockchain-based payment system, $\mathscr{C}$ needs to provide the corresponding deposits in $\mathscr{B}_{\mathscr{P}}$ as the cost of service before requesting $\mathscr{CS}$ to perform QR decomposition of large-scale matrix $\widehat{A}$. Meanwhile, $\mathscr{CS}$ also deposits the compensation in $\mathscr{B}_{\mathscr{P}}$ as a guarantee for honest computing. If outsourcing result is correct, $\mathscr{CS}$ can obtain the corresponding service fees from $\mathscr{B}_{\mathscr{P}}$. Otherwise, $\mathscr{C}$ informs $\mathscr{B}_{\mathscr{P}}$ to terminate the payment process, and $\mathscr{CS}$ will accept punishment and provide compensation to $\mathscr{C}$. In this paper, we focus on designing publicly verifiable outsourcing computation for QR decomposition scheme called $\mathscr{PVCMD}$-$\mathscr{QR}$.

$\mathscr{PVCMD}$-$\mathscr{QR}$ can be divided into five phases including initialization phase, encryption phase, computation phase, verification phase, and decryption phase. To better understand this process, a flowchart is shown in Figure 2. Now, the specific process of the scheme is described below.

In the initialization phase, $\mathscr{DC}$ runs the KeyGen algorithm. Here, it randomly generates three n-dimensional key vectors $\overrightarrow{s}$, $\overrightarrow{l}$, and $\overrightarrow{k}$ as the private key SK to produce the public key $\mathrm{PK} = (\mathrm{PK}_1, \mathrm{PK}_2, \mathrm{PK}_3)$ and the evaluation key EK. $\overrightarrow{s}$ and $\overrightarrow{l}$ are delivered to $\mathscr{C}$ and $\mathscr{V}$ through the secure

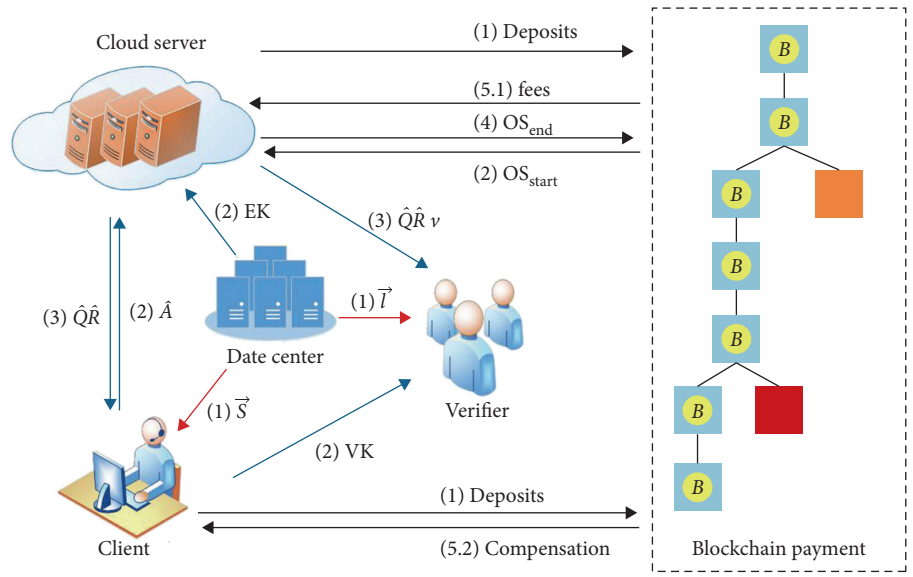

Figure 1: System model.

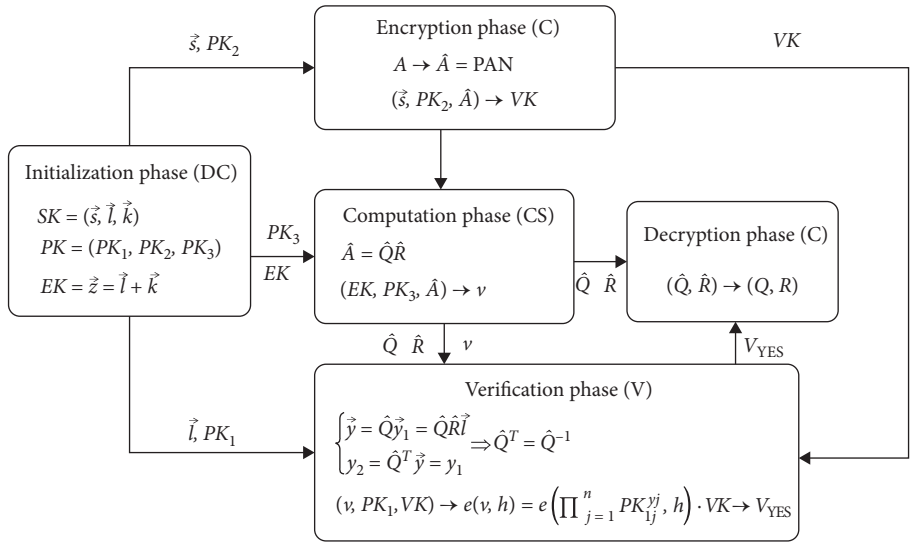

Figure 2: A plan flowchart of the proposed scheme.

channel, respectively. In the encryption phase, ProbGen algorithm is executed. $\mathscr{C}$ encrypts a full rank privacy matrix A into $\hat{A} = \text{PAN}$, $A \in Z_p^{n\times n}$, where block-diagonal upper triangular matrix N is constructed by $\mathscr{C}$. Meanwhile, the verification key VK is generated by PK_2, $\vec{s}$, and $\hat{A}$. Here, we make full use of the technique called matrix digest during the process of generating the VK in our scheme. By multiplying $\vec{s}$ and the encryption matrix $\hat{A}$ to obtain the vector $\vec{b}$, $\mathscr{V}$ uses PK_2 and $\vec{b}$ to create VK. Then, $\hat{A}$ is uploaded to $\mathscr{CS}$, and VK is provided to any $\mathscr{V}$ simultaneously. Then, the compute algorithm is implemented in the computation phase. $\mathscr{CS}$ receives $\hat{A}$ to perform QR decomposition. Using EK from $\mathscr{DC}$, it generates a value v for $\mathscr{V}$. After getting the orthogonal matrix $\hat{Q}$ and the upper triangular matrix $\hat{R}$, $\mathscr{V}$ begins to execute the verify algorithm by using both the key VK and the proof v in the verification phase. If the verification is true, V_{YES} is sent to $\mathscr{C}$ at once and $\mathscr{V}$ informs $\mathscr{C}$ to accept the decomposition results. By the matrix digest, the verifier uses the vector $\vec{l}$ and the decomposition results to produce the vector $\vec{y} = (y_1, \ldots, y_j, \ldots, y_n)$. It is this technology that prevents $\mathscr{V}$ from having to traverse each element of the result matrices. Eventually, in the decryption phase, utilizing the unit permutation matrix P^T and the inverse matrix N^{-1} of the matrix N, $\mathscr{C}$ runs the solve algorithm and decrypts the result matrices to get the orthogonal matrix Q and the upper triangular matrix R of A.

3.4. Specific Algorithms of $\mathscr{PVCMD}$-$\mathscr{QR}$. The $\mathscr{PVCMD}$-$\mathscr{QR}$ scheme consists of five subalgorithms, including KeyGen, ProbGen, Compute, Verify, and Solve.

Algorithm 1 (KeyGen) is executed by $\mathscr{DC}$.

Input:
$\mathcal{F}$, λ;
Output:
para and SK, PK, EK;
(1) Step 1: initialization
(2) compute para = $(p, G_1, G_2, G_T, g, h, \tilde{g}, \tilde{h})$.
(3) Step 2: generating keys
(4) generate SK and EK.
(5) compute PK
(6) Step 3: return (para, SK, PK, EK)

Algorithm 1: KeyGen algorithm.

There exist two cyclic groups G_1 and G_2 with the order p, where g is generator of G_1 and h is generator of G_2. So, a bilinear pairings can be described as $G_1 \times G_2 \longrightarrow G_T$. For any $\delta \in F_p^*$, it calculates $\tilde{h} = h^\delta$ and $\tilde{g} = g^\delta$. Afterwards, it publishes the parameter para = $(p, G_1, G_2, G_T, g, h, \tilde{g}, \tilde{h})$.

(a) It generates three vectors $\overrightarrow{s}$, $\overrightarrow{l}$, and $\overrightarrow{k}$ randomly, $s_i, l_i, k_i \in F_p^* (1 \le i \le n)$. Then, we regard these three vectors as the secret key SK. The algorithm uses the vectors $\overrightarrow{l}$ and $\overrightarrow{k}$ to determine the evaluation key EK = $\overrightarrow{z} = \overrightarrow{l} + \overrightarrow{k}$.

(b) By using SK = $(\overrightarrow{s}, \overrightarrow{l}, \overrightarrow{k})$, it produces the public key PK = $(\mathrm{PK}_1, \mathrm{PK}_2, \mathrm{PK}_3)$, which is defined as follows:

$$\begin{cases} \mathrm{PK}_1 = (\mathrm{PK}_{11}, \mathrm{PK}_{12}, \ldots, \mathrm{PK}_{1n}), \\ \mathrm{PK}_2 = (\mathrm{PK}_{21}, \mathrm{PK}_{22}, \ldots, \mathrm{PK}_{2n}), \\ \mathrm{PK}_3 = (\mathrm{PK}_{31}, \mathrm{PK}_{32}, \ldots, \mathrm{PK}_{3n}), \end{cases} \tag{5}$$

where $\mathrm{PK}_{1i} = g^{s_i}$, $\mathrm{PK}_{2i} = e(g^{k_i}, \tilde{h})$, and $\mathrm{PK}_{3i} = \tilde{g}^{s_i}$ for $i = 1$ to n.

Algorithm 2 (ProbGen)is expressed and is executed by $\mathcal{C}$ to encrypt a privacy matrix A.

(a) $\mathcal{C}$ needs to perform $\widehat{A} = \mathrm{PAN}$, where $N \in Z_p^{n \times n}$. In particular, the matrix P is an n-order unit permutation matrix, and the matrix N is a sparse block diagonal square matrix, whose main diagonal is composed of several matrices N_i and the remaining positions are all 0 elements:

$$N = \begin{pmatrix} N_1 & 0 & \cdots & 0 \\ 0 & N_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N_t \end{pmatrix}_{n \times n}, \tag{6}$$

where the submatrix N_i $(i = 1, 2, \ldots, t)$ is lower-order upper triangular invertible matrix, where $N_i \in Z_p^*$ is saved secretly.

(b) It uses the encryption matrix $\widehat{A}$ to obtain the verification key VK as below:

(i) The client uses $\overrightarrow{s}$ to produce the auxiliary vector $\overrightarrow{b}$, where $\overrightarrow{b} = \overrightarrow{s} \cdot \widehat{A}$.
(ii) PK_2 and $\overrightarrow{b}$ are used to generate VK, namely,

Input:
$\overrightarrow{s}$, PK_2, and A;
Output:
$\widehat{A}$ and VK
(1) Step 1: transforming matrix A
(2) Produce the matrices P and N.
(3) Encode A into $\widehat{A}$.
(4) Step 2: obtaining VK
(5) Compute $\overrightarrow{b}$.
(6) Obtain VK by equation (7).
(7) Step 3: return ($\widehat{A}$, VK)

Algorithm 2: ProbGen algorithm.

$$\mathrm{VK} = \prod_{i=1}^{n} \mathrm{PK}_{2i}^{b_i}. \tag{7}$$

In Algorithm 3 (Compute), $\mathcal{CS}$ conducts QR decomposition of $\widehat{A}$. The process of algorithm is shown in Algorithm 3.

(a) Breaking $\widehat{A}$ into $\widehat{A} = \widehat{Q}\widehat{R}$, $\mathcal{CS}$ performs operation of QR decomposition.

(b) It should calculate a proof v to prove the correctness of the decomposition results.

(i) An n-dimensional auxiliary vector $\overrightarrow{m}$ is generated, where $m_i = \prod_{j=1}^{n} \mathrm{PK}_{3j}^{\widehat{A}_{ji}}$ $(1 \le i \le n)$.
(ii) It uses the vector $\overrightarrow{m}$ and EK to generate the value $v = \prod_{i=1}^{n} m_i^{z_i}$.

Algorithm 4 (Verify) is conducted after $\mathcal{V}$ receives the information from other participants. The specific process is explained in Algorithm 4.

(a) $\mathcal{V}$ should inspect the orthogonality of $\widehat{Q}$.

(i) $\mathcal{V}$ uses $\overrightarrow{l}$ and $\widehat{R}$ to generate a intermediate vector $\overrightarrow{y}_1$. By multiplying $\widehat{Q}$ by the column vector $\overrightarrow{y}_1$, it can generate a result vector $\overrightarrow{y}$, which is carried out in field of real number, namely,

$$\begin{aligned} \overrightarrow{y}_1 &= \widehat{R}\,\overrightarrow{l}, \\ \overrightarrow{y} &= \widehat{Q}\,\overrightarrow{y}_1. \end{aligned} \tag{8}$$

Input:
$\widehat{A}$, PK_3 and EK;
Output:
$\widehat{Q}$, $\widehat{R}$ and v
(1) Step 1: QR decomposition of $\widehat{A}$
(2) Decompose $\widehat{A}$ into $\widehat{Q}$ and $\widehat{R}$.
(3) Step 2: obtaining v
(4) Generate $\overrightarrow{m}$.
(5) Compute v with (EK, $\overrightarrow{m}$).
(6) Step 3: return $\widehat{Q}$, $\widehat{R}$, and v

Algorithm 3: Compute algorithm.

Input:
$(\widehat{Q}, \widehat{R})$, v, $\overrightarrow{l}$, PK_1 and VK;
Output:
V_{YES} or V_{NO}
(1) Step 1: checking the orthogonality of $\widehat{Q}$
(2) Get $\overrightarrow{y}_1$ with $(\widehat{R}, \overrightarrow{l})$.
(3) Produce $\overrightarrow{y}$ with $(\widehat{Q}, \overrightarrow{y}_1)$.
(4) Compute $\overrightarrow{y}_2$ with $(\widehat{Q}^T, \overrightarrow{y})$.
(5) $\overrightarrow{y}_2 \stackrel{?}{-} \overrightarrow{y}_1 = 0$.
(6) Step 2: checking the correctness of $\widehat{Q}$ and $\widehat{R}$
(7) Compute $e(v, h)$.
(8) Compute $e(\prod_{j=1}^{n} PK_{1j}^{y_j}, \widetilde{h}) \cdot VK$.
(9) $e(v, h) = e(\prod_{j=1}^{n} PK_{1j}^{y_j}, \widetilde{h}) \cdot VK \longrightarrow V_{YES}$
(10) $e(v, h) \neq e(\prod_{j=1}^{n} PK_{1j}^{y_j}, \widetilde{h}) \cdot VK \longrightarrow V_{NO}$
(11) Step 3: return V_{YES} or V_{NO}

Algorithm 4: Verify algorithm.

(ii) It multiplies the vector $\overrightarrow{y}$ by $\widehat{Q}^T$ to obtain a new vector denoted by $\overrightarrow{y}_2$, where $\widehat{Q}^T$ is transpose of matrix $\widehat{Q}$. Due to property of orthogonal matrix, if $\overrightarrow{y}_2 = \widehat{Q}^T\overrightarrow{y} = \widehat{Q}^T\widehat{Q}\overrightarrow{y}_1 = \overrightarrow{y}_1$ is true, next step (b) is executed.

(b) If the following equation holds in the finite field, $\mathscr{V}$ outputs V_{YES}. Otherwise, it outputs V_{NO}:

$$e(v, h) = e\left(\prod_{j=1}^{n} PK_{1j}^{y_j}, \widetilde{h}\right) \cdot VK. \tag{9}$$

Suppose that the nonsingular matrix A can be decomposed into $A = QR$. Algorithm 5 (Solve) is executed by $\mathscr{C}$ to obtain both Q and R.

(a) $\mathscr{C}$ gets the transposed matrix P^T and the inverse matrix N^{-1} of N which needs to solve the inverse of the upper triangular submatrix N_i for $i = 1$ to t.

(b) Multiplying P^T by the left of $\widehat{Q}$ and N^{-1} by the right of $\widehat{R}$, the QR decomposition of matrix A can be obtained:

$$\begin{cases} Q = P^T \cdot \widehat{Q}, \\ R = \widehat{R} \cdot N^{-1}. \end{cases} \tag{10}$$

Input:
$\widehat{Q}$ and $\widehat{R}$;
Output:
Q and R
(1) Solve P^T and N^{-1}.
(2) Obtain Q and R by equation (10).
(3) Return Q and R.

Algorithm 5: Solve algorithm.

4. Protocol Analysis

In this section, $\mathscr{PVCMD}$-$\mathscr{QR}$ is analyzed from the perspectives of correctness, security, and efficiency.

4.1. Correctness Analysis

4.1.1. ProbGen Algorithm. Since the result of QR decomposition is unique when the main diagonal elements are positive in the upper triangular matrix, not all matrices can be decomposed and the square matrix to be decomposed must be invertible and nonsingular. Therefore, conditions of decomposition of the input matrix $\widehat{A}$ should satisfy $|\widehat{A}| \neq 0$.

In fact, after the matrix A is encrypted, this condition is still satisfied. From $\widehat{A} = PAN$, we get the equation $|\widehat{A}| = |P| \cdot |A| \cdot |N|$. Specifically, since both P and N are invertible matrices, $|P| \neq 0$ and $|N| \neq 0$. In addition, the privacy matrix A is a full rank matrix, $|A| \neq 0$.

4.1.2. Verify Algorithm. Considering the property of the orthogonal matrix, there is $Q^TQ = I$, where I represents the identity matrix. If the vectors $\overrightarrow{y}_2$ and $\overrightarrow{y}_1$ are identical, $\widehat{Q}$ must be an orthogonal matrix. However, the matrix $\widehat{R}$ can be observed directly. The results are correct if the parties involved in the scheme execute the agreement honestly.

Before verification, it is necessary for $\mathscr{V}$ to compute the result vector $\overrightarrow{y}$, which can be obtained by $\overrightarrow{y} = \widehat{Q}(\widehat{R}\overrightarrow{l})$.

Because of equations (11) and (12), equation (13) holds

$$\overrightarrow{b} \cdot \overrightarrow{l} = (\overrightarrow{s}\widehat{A})\overrightarrow{l} = \overrightarrow{s}(\widehat{Q}\widehat{R}\overrightarrow{l}) = \overrightarrow{s} \cdot \overrightarrow{y}, \tag{11}$$

$$b_i = \sum_{j=1}^{n} s_j \widehat{A}_{ji}, \quad 1 \leq i \leq n, \tag{12}$$

$$\begin{aligned} \sum_{i=1}^{n} b_i l_i &= \sum_{i=1}^{n} \left(\sum_{i=1}^{n} s_j \widehat{A}_{ji} \right) l_i \\ &= \sum_{i=1}^{n} s_j \left(\sum_{i=1}^{n} \widehat{Q}_{ji} \sum_{i=1}^{n} \widehat{R}_{ik} l_k \right) = \sum_{i=1}^{n} s_j y_j. \end{aligned} \tag{13}$$

According to equation (13), we have

$$
\begin{aligned}
e(\mathbf{v}, h) &= e\left(\prod_{i=1}^{n} m_i^{z_i}, h\right) = e\left(\prod_{i=1}^{n}\left(\prod_{j=1}^{n} \mathrm{PK}_{3j}^{\widehat{A}_{ji}}\right)^{z_i}, h\right) = e\left(\prod_{i=1}^{n}\left(\prod_{j=1}^{n} \widetilde{g}^{s_j \widehat{A}_{ji}}\right)^{z_i}, h\right) \\
&= e\left(\prod_{i=1}^{n} g^{\delta\left(\sum_{j=1}^{n} s_j \widehat{A}_{ji}\right) z_i}, h\right) = e\left(\prod_{i=1}^{n} g^{\delta b_i z_i}, h\right) = e\left(\prod_{i=1}^{n} g^{b_i l_i}, \widetilde{h}\right) e\left(\prod_{i=1}^{n} g^{b_i k_i}, \widetilde{h}\right) \\
&= e\left(g^{\sum_{j=1}^{n} s_j y_j}, \widetilde{h}\right) \prod_{i=1}^{n} e\left(g^{k_i}, \widetilde{h}\right)^{b_i} = e\left(\prod_{j=1}^{n} \mathrm{PK}_{1j}^{y_j}, \widetilde{h}\right) \cdot \mathrm{VK}.
\end{aligned} \tag{14}
$$

In short, equation (9) is established and the verification is successful and sound.

4.2. Security Analysis

Theorem 1. *The publicly verifiable computation scheme $\mathcal{PVCMD}$-$\mathcal{QR}$ is secure under the co-CDH in group G_1.*

Proof. We follow Definition 3 to illustrate the theoretical analysis for security of our proposed scheme.

To prove this theorem, there are two adversaries $\mathcal{A}$ and $\mathcal{B}$. Suppose that the adversary $\mathcal{A}$ has a very strong ability to destroy the soundness of $\mathcal{PVCMD}$-$\mathcal{QR}$ with a probability ε so that it can obtain important information of the scheme. However, the challenger $\mathcal{B}$ with these information from adversary $\mathcal{A}$ tries the best to address the co-CDH problem with a nonnegligible probability ε', and $\varepsilon \approx \varepsilon'$.

To break this assumption, challenger $\mathcal{B}$ accesses the random oracle $O_{\text{co-CDH}}$ which generates $g, g' = g^{\alpha} \in G_1$ and $h, h' = h^{\beta} \in G_2$ as the result of output in return and selects $\alpha, \beta \in F_P^*$.

Then, the challenger $\mathcal{B}$ simulates adversary $\mathcal{A}$ to carry out this soundness experiment:

Adversary $\mathcal{B}$ denotes $\widetilde{g}' = g'^{\delta}$ and $\widetilde{h}' = h'^{\delta}$ by selecting $\delta \in F_p^*$ randomly as well as generates the parameter para$'$, namely, para$' = (p, G_1, G_2, G_T, e, g, \widetilde{g}', h, \widetilde{h}')$. It uses parameters para$'$ to generate PK_1' and PK_3' respectively, which are shown as follows:

$$
\mathrm{PK}_1' = \left(\mathrm{PK}_{11}', \mathrm{PK}_{12}', \ldots, \mathrm{PK}_{1n}'\right), \tag{15}
$$

where $\mathrm{PK}_{1i}' = g'^{s_i}$, $(1 \le i \le n)$.

$$
\mathrm{PK}_3' = \left(\mathrm{PK}_{31}', \mathrm{PK}_{32}', \ldots, \mathrm{PK}_{3n}'\right), \tag{16}
$$

where $\mathrm{PK}_{3i}' = g'^{s_i}$, $(1 \le i \le n)$.

Then, it generates an auxiliary vector $m' \in G_1^{n \times 1}$, where $m_i' = \prod_{j=1}^{n} \mathrm{PK}_{3j}^{'\widehat{A}_{ji}}$ for i to n.

According to m' and PK_1', it computes the public key PK_2' which is a vector. In other words, the expression of PK_2' is determined directly:

$$
\mathrm{PK}_2' = \left(\mathrm{PK}_{21}', \mathrm{PK}_{22}', \ldots, \mathrm{PK}_{2n}'\right). \tag{17}
$$

We take each element of PK_2' as the following:

$$
\mathrm{PK}_{2i}' = \left(\frac{e\left(m_i^{'z_i}, h\right)}{e\left(\mathrm{PK}_{1i}^{'y_i}, \widetilde{h}'\right)}\right)^{(1/b_i)} = \left(\frac{e\left(\left[\prod_{j=1}^{n} \mathrm{PK}_{3j}^{'\widehat{A}_{ji}}\right]^{z_i}, h\right)}{e\left(\mathrm{PK}_{1i}^{'y_i}, \widetilde{h}'\right)}\right)^{(1/b_i)}. \tag{18}
$$

There is an important condition for the above expression to be correct, namely, $b_i \neq 0$. Since the matrix $\widehat{A}$ is full rank and $\overrightarrow{s} \in F_p^*$, the vector $\overrightarrow{b} = \overrightarrow{s} \cdot \widehat{A}$ is a nonzero vector. In addition, it defines evaluation key $\mathrm{EK}' = (z_1, \ldots, z_i, \ldots, z_n)$.

Therefore, the challenger $\mathcal{B}$ can obtain some corresponding information eventually to complete this experiment such as PK_2', para$'$, and EK'.

Secondly, different from the real output of the KeyGen algorithm, the distribution of the output of the random oracle O_{keyGen} is independent and indistinguishable. So, we have reason to believe these two facts:

(a) According to the vectors $\overrightarrow{b}$ and $\overrightarrow{y} = \widehat{Q}(\widehat{R}\,\overrightarrow{l})$, the following formula must be correct:

$$
e\left(\prod_{i=1}^{n} m_i^{'z_i}, h\right) = e\left(\prod_{j=1}^{n} \mathrm{PK}_{1j}^{'y_j}, \widetilde{h}'\right) \cdot \prod_{i=1}^{n} \mathrm{PK}_{2i}^{'b_i}. \tag{19}
$$

(b) The vector $\overrightarrow{m}$ and the key $\mathrm{PK} = (\mathrm{PK}_1, \mathrm{PK}_2, \mathrm{PK}_3)$ are identical to the statistically distribution of $\overrightarrow{m'}$ and $\mathrm{PK}' = (\mathrm{PK}_1', \mathrm{PK}_2', \mathrm{PK}_3')$ severally.

Then, adversary $\mathcal{A}$ takes advantage of $\widehat{A}$ and PK_2' to access the random oracle O_{ProbGen}. The attacker $\mathcal{B}$ imitates O_{ProbGen} and takes the matrix $\widehat{A}$ and VK' as output, where we denote $\mathrm{VK}' = \prod_{i=1}^{n} \mathrm{PK}_{2i}^{'b_i}$.

Finally, adversary $\mathcal{A}$ will return a value v and fake results $(\widehat{Q}^*, \widehat{R}^*)$ with $\widehat{Q}^* \widehat{R}^* \neq \widehat{A}$. Then, challenger $\mathcal{B}$ will calculate the result vector $\overrightarrow{y}^* = \widehat{Q}^*(\widehat{R}^* \overrightarrow{l})$ and verify whether $\overrightarrow{y}^* = \overrightarrow{y}$ is true. If the invalid result $\overrightarrow{y}^*$ passes this verification, it means that challenger $\mathcal{B}$ has failed and does not break the assumption. Otherwise, it returns the following expression

of $g^{\alpha\beta}$ and declares that the difficult assumption co-CDH is broken, while it is impractical:

$$g^{\alpha\beta} = \left(\frac{v}{\prod_{i=1}^{n} m_i^{'z_i}}\right)^{\left[\delta\sum_{j=1}^{n} s_j\left(y_j^* - y_j\right)\right]^{-1}} = \left(\frac{v}{\prod_{i=1}^{n}\left[\prod_{j=1}^{n} \mathrm{PK}_{3j}^{'\widehat{A}_{ji}}\right]^{z_i}}\right)^{\left[\delta\sum_{j=1}^{n} s_j\left(y_j^* - y_j\right)\right]^{-1}}. \tag{20}$$

Now, we are going to provide the specific process of the above solving $g^{\alpha\beta}$.

If the wrong vector $\overrightarrow{y}^*$ passes the verification, the following equation (21) must be true:

$$e(v,h) = e\left(\prod_{j=1}^{n} \mathrm{PK}_{1j}^{'y_j^*}, h'\right)\cdot \mathrm{VK}'. \tag{21}$$

Considering equation (18), it is achieved that

$$e\left(\prod_{i=1}^{n} m_i^{'z_i}, h\right) = e\left(\prod_{j=1}^{n} PK_{1j}^{'y_j}, \widetilde{h}'\right)\cdot \mathrm{VK}'. \tag{22}$$

To divide equations (21) with (22), we obtain

$$\begin{aligned} e\left(\frac{v}{\prod_{i=1}^{n} m_i^{'z_i}}, h\right) &= e\left(\prod_{j=1}^{n} \mathrm{PK}_{1j}^{'\left(y_j^* - y_j\right)}, \widetilde{h}'\right) \\ &= e\left(\prod_{j=1}^{n} g^{'s_j\left(y_j^* - y_j\right)}, \widetilde{h}'\right). \end{aligned} \tag{23}$$

As $g' = g^{\alpha}, \widetilde{h}' = h^{\delta\beta}$, and $m_i' = \prod_{j=1}^{n} \mathrm{PK}_{3j}^{'\widehat{A}_{ji}}$, it is concluded that

$$\begin{aligned} e\left(\frac{v}{\prod_{i=1}^{n} m_i^{'z_i}}, h\right) &= e\left(\prod_{j=1}^{n} g^{\alpha s_j\left(y_j^* - y_j\right)}, \widetilde{h}'\right) \\ &= e\left(g^{\alpha\beta\sum_{j=1}^{n}\delta s_j\left(y_j^* - y_j\right)}, h\right). \end{aligned} \tag{24}$$

Hence, if $(y_j^* - y_j) \neq 0$ and $\delta s_j \neq 0$, there is

$$g_{j=1}^{\alpha\beta\sum n\delta s_j\left(y_j^* - y_j\right)} = \frac{v}{\prod_{i=1}^{n} m_i^{'z_i}}, \tag{25}$$

namely,

$$g^{\alpha\beta} = \left(\frac{v}{\prod_{i=1}^{n}\left[\prod_{j=1}^{n} \mathrm{PK}_{3j}^{'\widehat{A}_{ji}}\right]^{z_i}}\right)^{\left[\delta\sum_{j=1}^{n} s_j\left(y_j^* - y_j\right)\right]^{-1}}. \tag{26}$$

Hence, if this scheme is destroyed by adversary $\mathcal{A}$ with a certain probability ε, challenger $\mathcal{B}$ is able to break the co-CDH with a nonnegligible advantage ε'. In summary, $\mathcal{PVCMD}$-$\mathcal{QR}$ is secure under the co-CDH in group G_1.

4.3. Efficiency Analysis. In this section, we intend to give a detailed analysis of the computational overhead of $\mathcal{PVCMD}$-$\mathcal{QR}$.

The matrix N, where the order of each submatrix N_i for $i = 1$ to t is chosen randomly from 2 to w ($w \ll n$), is produced by $\mathcal{C}$, so there will be many combinations in reality. However, since the matrix N is sparse with the computational complexity $O(n^2)$ for solving N^{-1}, the computational overhead is not taken into consideration about generation of N and N^{-1}.

To simplify the analysis, suppose that each submatrix N_i is a w-order upper triangular matrix in the main diagonal of N. However, the inverse matrix of the w-order upper triangular matrix is obtained easily, so it is convenient to obtain the inverse matrix N^{-1} of N, where the inverse N_i^{-1} of submatrix N_i ($1 \le i \le t$) is placed in the corresponding position, like this

$$N^{-1} = \begin{pmatrix} N_1^{-1} & 0 & \cdots & 0 \\ 0 & N_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N_t^{-1} \end{pmatrix}_{n\times n}. \tag{27}$$

Therefore, we suppose that the order n of the original matrix A should meet this condition, namely, $n = wt$.

In KeyGen algorithm, three vectors $\overrightarrow{s}$, $\overrightarrow{l}$, and $\overrightarrow{k}$ which are generated randomly require $3n$ random numbers in the group operation. Next, it needs to calculate PK_1, PK_2, PK_3, and EK separately. The public key PK_1 is an n-dimensional vector where there is $\mathrm{PK}_{1i} = g^{s_i}$, so $\mathcal{DC}$ will execute n exponential operations to obtain PK_1. Since the public key PK_2, an n-dimensional vector, is obtained by n exponential operations and n pairing operations similarly, where $\mathrm{PK}_{2i} = e(g^{k_i}, \widetilde{h})$, additionally, n exponential operations needs to be performed to get the public key PK_3. As the evaluation key EK is also an n-dimensional vector, $\mathcal{DC}$ should perform n additions.

To perform ProbGen algorithm where we have $\widehat{A} = \mathrm{PAN}$, $\mathcal{C}$ needs to use a sparse block diagonal upper triangular matrix N and a unit permutation matrix P. Therefore, there are $n^2 + (1/2)(w+1)n^2$ multiplications and $(1/2)(w-1)n^2$ additions in encryption operation. On the contrary, $\mathcal{C}$ also computes a verification key $\mathrm{VK} = \prod_{i=1}^{n} \mathrm{PK}_{2i}^{b_i}$ with the vector $\overrightarrow{b}$. To get the vector, it is going to perform n^2 multiplications. Therefore, both n exponentials and $n-1$ multiplications should be required in the process of generating VK.

$\mathcal{CS}$ executes the QR decomposition of the matrix $\widehat{A}$ according to Compute algorithm. It is necessary to produce the value v, where this process involves n exponentiation operations and $n-1$ multiplications. However, before generating the value v, it should utilize the public key PK_3 to get an n-dimensional auxiliary vector $\overrightarrow{m}$, which requires n^2 exponential and $n(n-1)$ multiplications operations.

Table 1: Computation cost of each phase in $\mathcal{PVCMD}$-$\mathcal{QR}$.

Algorithm	Computation cost
KeyGen	$3n\text{Ex} + n\text{Pa} + n\text{Ad} + 3n\text{Ge}$
ProbGen	$[(1/2)(w+5)n^2 + n - 1]\text{Mu} + n\text{Ex} + (1/2)(w-1)n^2\text{Ad}$
Compute	$\text{De} + (n^2-1)\text{Mu} + (n^2+n)\text{Ex}$
Verify	$(1/2)(5n^2+3n)\text{Mu} + n\text{Ex} + (3/2)(n^2-n)\text{Ad} + 2\text{Pa}$
Solve	$[(1/4)(w+5)n^2 + T_1 n]\text{Mu} + [(1/4)(w-1)n^2 + T_2 n]\text{Ad}$

Table 2: Computation cost of $\mathcal{PVCMD}$-$\mathcal{QR}$ scheme for different problem sizes.

Dimension	KeyGen (ms)	ProbGen (ms)	Compute (ms)	Verify (ms)	Solve (ms)
$n = 400$	33.910000	13512.718000	167927.372000	88.412800	11574.687000
$n = 500$	43.037000	28093.517000	264634.147000	139.749000	22764.303000
$n = 600$	52.619000	55298.470000	377979.489000	203.310000	39072.307000
$n = 700$	62.410000	97193.756000	513340.065000	278.004000	61935.196000
$n = 800$	70.836000	159190.287000	670820.937000	365.882000	92610.783000

In Verify algorithm, $\mathcal{V}$ must generate an n-dimensional result vector $\overrightarrow{y}$, firstly, where $(1/2)n(n+1) + n^2$ multiplications and $(1/2)(n-1)n + n(n-1)$ additions should be performed. When checking the orthogonal property of the matrix $\widehat{Q}$, $\mathcal{V}$ is asked to compute n^2 multiplications. Then, it takes advantage of v, $\overrightarrow{y}$, and VK to verify whether $e(v,h) = e(\prod_{j=1}^{n} \text{PK}_{1j}^{y_j}, \widetilde{h}) \cdot \text{VK}$ holds, which needs n exponentiation operations, $n-1+1$ multiplications, and two pairing operations in this phases.

We also should take the computation cost of *Solve* algorithm. $\mathcal{C}$ has to decrypt the matrices $\widehat{Q}$ and $\widehat{R}$ to obtain the result of QR decomposition of original matrix A. Therefore, n^2 multiplication operations are carried out for solving $Q = P^T\widehat{Q}$. In order to compute $R = \widehat{R}N^{-1}$, it will deal with $(1/4)(w+1)n^2 + ((1/4)w + (1/3) - (1/12)w^2)n$ multiplications and $(1/4)(w-1)n^2 + ((1/4)w + (1/3) - (1/12)w^2)n$ additions.

Here, we denote an exponentiation operation with Ex, a multiplication operation with Mu, an addition operation with Ad, a pairing operation with Pa, a matrix decomposition with De, and a random number generation operation with Ge. T_1 and T_2 are described as follows:

$$\begin{cases} T_1 = \dfrac{1}{4}w + \dfrac{1}{3} - \dfrac{1}{12}w^2, \\ \\ T_2 = \dfrac{1}{4}w - \dfrac{1}{6} - \dfrac{1}{12}w^2, \end{cases} \tag{28}$$

where $w \ll n$.

In summary, the computation cost of each algorithm is shown in Table 1.

According to the above analysis, the computational complexity of the client is $O(n^2)$ and is lower than to accomplish QR decomposition directly.

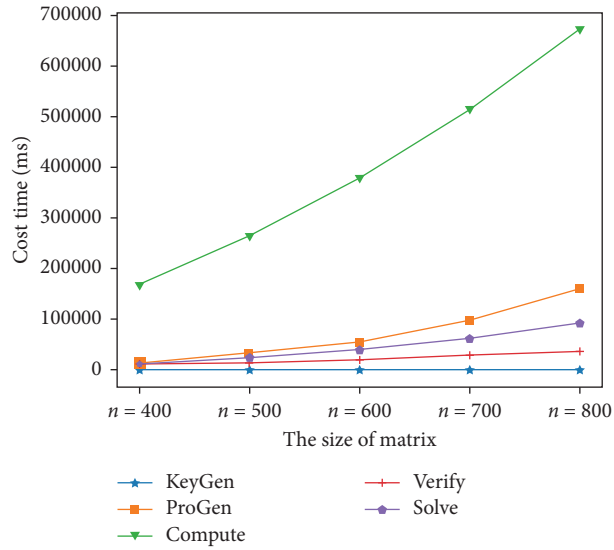

Figure 3: Computational time cost for each algorithm.

4.4. Experiment Analysis. Here, we evaluate the proposed scheme with experiments. Using C language, we emulate the data center DC, the client C, the cloud server CS, and the verifier V on a laptop with Intel Core(TM) i5-8265U CPU processor, 8 GB RAM memory.

To better describe the computational efficiency of the proposed $\mathcal{PVCMD}$-$\mathcal{QR}$ scheme, we simulate all these algorithms in our scheme (i.e., KeyGen, ProbGen, Compute, Verify, and Solve). First, we assume that the order of each submatrix of the block diagonal matrix N is identical, $w = 25$. The computation costs with different scales of the problem are listed in Table 2, and the specific trend is shown in Figure 3. The experiment shows that the overhead of the client side is smaller than the CS, as listed in Table 3.

Table 3: Comparison of computation cost between the client and cloud side.

Dimension	Client cost in $\mathcal{PVCMD}$-$\mathcal{QR}$ (ms)	Cloud server cost in $\mathcal{PVCMD}$-$\mathcal{QR}$ (ms)
$n = 400$	25175.817800	167927.372000
$n = 500$	50997.569000	264634.147000
$n = 600$	94574.087000	377979.489000
$n = 700$	159406.956000	513340.065000
$n = 800$	252166.952000	670820.937000

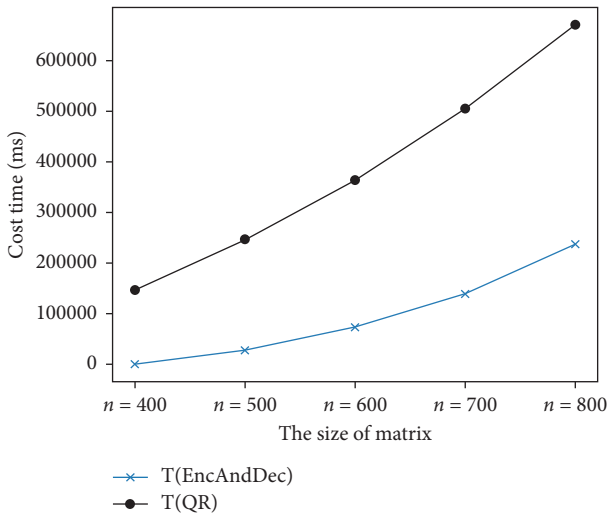

Figure 4: Efficiency comparison between T(EncAndDec) and T(QR).

Then, we illustrate the superiority of the outsourcing computation in Figure 4, in which we mainly consider the time cost of $\mathcal{C}$. In Figure 4, the symbol T(EncAndDec) represents the time cost of $\mathcal{C}$ in encryption and decryption phases of outsourcing process, and the symbol T(QR) means the time cost is required for the $\mathcal{C}$ to compute QR decomposition of matrix A directly. Compared to directly computing QR decomposition on the original matrix A, the $\mathcal{PVCMD}$-$\mathcal{QR}$ scheme is more efficient obviously as the dimension of matrix increases.

5. Conclusion

Aiming at the public verification outsourcing computation, this paper proposes a new publicly verifiable scheme with blockchain payment under the amortized model for QR decomposition of large-scale matrix. The sensitive data information is protected by using the sparse matrix. Therefore, client can upload his/her privacy matrix to the outsourcing service provider to perform QR decomposition. Simultaneously, the matrix digest technique is applied to the verification operation of outsourcing computation, which cuts down the workload of verifier dramatically. Afterwards, we also provide the specific theoretical proof of the correctness, safety, and efficiency of the $\mathcal{PVCMD}$-$\mathcal{QR}$ scheme, and the result proves that the scheme is secure under the co-CDH assumption.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation of China under Grant nos. 61902314, 62072371, and 61772418, Natural Science Basic Research Plan in Shaanxi Province of China under Grant no. 2018JZ6001, and Basic Research Program of Qinghai Province under Grants no. 2020-ZJ-701.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, and K.-K. R. Choo, "Cryptcloud+: secure and expressive data access control for cloud storage," *IEEE Transactions on Services Computing*, vol. 14, pp. 111–124, 2021.

[3] J. Ning, X. Huang, W. Susilo, K. Liang, X. Liu, and Y. Zhang, "Dual access control for cloud-based data storage and sharing," *IEEE Transactions on Dependable and Secure Computing*, no. 99, p. 1, 2020.

[4] R. Chow, P. Golle, M. Jakobsson et al., "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pp. 85–90, ACM, Chicago, IL, USA, November 2009.

[5] Z. Yinghui, D. Robert, L. Ximeng, and Z. Dong, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Transactions on Services Computing*, p. 1, 1939.

[6] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, pp. 169–178, Bethesda, MD, USA, May 2009.

[7] X. Chen, "Introduction to secure outsourcing computation," *Synthesis Lectures on Information Security, Privacy, & Trust*, pp. 1–93, Morgan & Claypool, San Rafael, CA, USA, 2016.

[8] 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.

[9] S. Salinas, C. Luo, X. Chen, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale sparse linear systems of equations," *IEEE Transactions on Big Data*, vol. 4, no. 1, pp. 26–39, 2018.

[10] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 69–78, 2015.

[11] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Information Sciences*, vol. 280, pp. 205–217, 2014.

[12] K. Jia, H. Li, D. Liu, and S. Yu, "Enabling efficient and secure outsourcing of large matrix multiplications," in *Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, San Diego, CA, USA, December 2015.

[13] S. Zhang, H. Li, Y. Dai, J. Li, M. He, and R. Lu, "Verifiable outsourcing computation for matrix multiplication with improved efficiency and applicability," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5076–5088, 2018.

[14] S. Zhang, C. Tian, H. Zhang, J. Yu, and F. Li, "Practical and secure outsourcing algorithms of matrix operations based on a novel matrix encryption method," *IEEE Access*, vol. 7, pp. 53823–853838, 2019.

[15] P. Golle and I. Mironov, "Uncheatable distributed computations," in *Cryptographers Track at the RSA Conference*, vol. 2020, pp. 425–440, Springer, Berlin, Germany, 2001.

[16] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Annual Cryptology Conference*, pp. 111–131, Springer, Berlin, Germany, 2011.

[17] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," *IACR Cryptology ePrint Archive*, vol. 2012, p. 281, 2012.

[18] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: verifiable computation from attribute-based encryption," in *Proceedings of the Theory of Cryptography Conference*, pp. 422–439, Taormina, Italy, March 2012.

[19] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pp. 160–164, IEEE, Washington, DC, USA, November 1982.

[20] K. Elkhiyaoui, M. Önen, M. Azraoui, and R. Molva, "Efficient techniques for publicly verifiable delegation of computation," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 119–128, Xi'an, China, May 2016.

[21] H. Li, S. Zhang, T. H. Luan, H. Ren, Y. Dai, and L. Zhou, "Enabling efficient publicly verifiable outsourcing computation for matrix multiplication," in *Proceedings of the 2015 International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 44–50, IEEE, Sydney, Australia, November 2015.

[22] X. Zhang, T. Jiang, K.-C. Li, A. Castiglione, and X. Chen, "New publicly verifiable computation for batch matrix multiplication," *Information Sciences*, vol. 479, pp. 664–678, 2019.

[23] O. E. Bronlund and T. L. Johnsen, "QR-factorization of partitioned matrices: solution of large systems of linear equations with non-definite coefficient matrices," *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 153–172, 1974.

[24] S. Li, J. Wen, F. Luo, T. Cheng, and Q. Xiong, ""A location and reputation aware matrix factorization approach for personalized quality of service prediction," in *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*, pp. 652–659, IEEE, Honolulu, HI, USA, June 2017.

[25] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for QOS prediction in service selection," in *Proceedings of the 2012 IEEE Ninth International Conference on Services Computing*, pp. 162–169, IEEE, Honolulu, HI, USA, June 2012.

[26] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QOS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, 2017.

[27] 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.

[28] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: outsourcing computation to untrusted workers," in *Annual Cryptology Conference*, pp. 465–482, Springer, Berlin, Germany, 2010.

[29] G. Sheng, C. Tang, W. Gao, and Y. Yin, "Md-$\mathscr{VC}_{\text{Matrix}}$: an efficient scheme for publicly verifiable computation of outsourced matrix multiplication," in *Proceedings of the International Conference on Network and System Security*, pp. 349–362, Springer, Taipei, Taiwan, September 2016.

[30] S. Micali, O. Goldreich, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth ACM Symposium on Theory of Computing, STOC*, pp. 218–229, New York, NY, USA, January 1987.