Efficient and Publicly Verifiable Outsourcing of Large-scale Matrix Multiplication

Gang Sheng[1], Chunming Tang[1], Wei Gao[2], Ying Yin[3], Yunlu Cai[1]

[1] College of Mathematics and Information Science, Guangzhou University, China
[2] School of Mathematics and Statistics Science, Ludong University, China
[3] College of Computer Science and Engineering, Northeastern University, China
shenggang@neusoft.edu.cn, ctang@gzhu.edu.cn, gaow@ldu.edu.cn, yingying@cc.neu.edu.cn, yunlucai@gzhu.edu.cn

Abstract

Matrix multiplication is an important operation, whose computation overhead is large for the matrix with big size. The researchers seek to delegate the computation to the cloud service provider with abundant resources. The security issues arise because the user loses direct control on the data, such as privacy preservation, result verification, etc. We investigate the problem of publicly verifiable matrix multiplication, where the third party verifier can verify the correctness of the returned result from the service provider with public key. The state-of-the-art schemes work inefficiently in practice because a number of computationally expensive operations are utilized for the purpose of public verification. We introduce the notion of matrix digest, on which an efficient scheme MD- $\mathcal{VC}_{Matrix}$ is presented. A one-dimensional vector is used for the verification-related computing, which is inverted from the original two-dimensional matrix. The computing on the verification-related computing is decreased significantly, thus the running efficiency of the related algorithms is promoted. We further present a fast algorithm for computing the batch of exponentiations. The security analysis demonstrates the security of our proposed outsourcing scheme, and the performance analysis shows the running efficiency of the scheme.

Keywords: Cloud computing, Outsourced computation, Public verification, Matrix multiplication

1 Introduction

The matrix is a common tool in higher algebra, which is widely used in statistical analysis, circuit theory, 3D animation, etc. Matrix has a wealth of computing forms, such as multiplication, eigenvalue, matrix decomposition, and so on. Matrix plays a very important role in the current scientific research and industrial production.

The scale of the matrix is very large in many scenarios. Although the computer is deployed with high computing speed CPU now, matrix operations still need to consume a large amount of computing resources. Therefore, high-performance equipment is needed to be configured, which is not a small overhead for some research institutions and researchers. Furthermore, it needs to spend a lot of manpower and material resources to carry out day-to-day management and maintenance. Fortunately, with the maturity and popularity of cloud computing technology and big data platform, large-scale computing tasks can be entrusted to cloud computing or big data platform with rich resources.

However, the cloud service provider or the big data platform is not entirely trusted. When the computing task is outsourced to the cloud computing service provider or the big data platform, the data owner loses direct control over the data, and thus some security issues may occur, including data leakage and incorrect results.

We study the problem of publicly verifiable computation of outsourced matrix multiplication. Many secure schemes have already been proposed [1-3]. Fiore et al. proposed a secure and publicly verifiable scheme $\mathcal{VC}_{Matrix}$ for outsourced matrix multiplication, which is deployed in the amortized model [1]. Fiore et al. makes full use of the closed-form efficiency property of the new pseudo-random functions to improve the processing efficiency on the user side. Li et al. followed the framework of [1] to propose a new scheme [2], where the user gets only an element in the ProbGen stage, the running efficiency is improved correspondingly.

However, we observe that, in the state-of-the-art schemes, a large number of computationally expensive computations are involved. For example, the verification element $w_{i,j} = g^{\alpha m_{i,j}} \cdot F_K(i,j)$ is computed for each element $m_{i,j}$ of the matrix M in the

*Corresponding Author: Chunming Tang; E-mail: ctang@gzhu.edu.cn
DOI: 10.3966/160792642018081904027

algorithm KeyGen. The verification object $VO=(V_1,\ldots,V_n)$ with $V_j=\prod_{i=1}^{n} w_{i,j}^{x_j}$ is computed in the algorithm Compute.

Both KeyGen and Compute work with time complexity $O(n^2)$ in the group domain, which is especially computationally expensive. Thus, the running efficiency of the existing publicly verifiable schemes for outsourced matrix multiplication is practically not so high. The schemes work inefficiency owning to the adoption of computationally expensive computations in the verification-related computing.

In this paper, we propose a scheme, called MD-$\mathcal{VC}_{Matrix}$, for the problem of publicly verifiable outsourcing for large-scale matrix multiplication. On the basis of the algebraic properties of matrix, the Matrix digest (MD) is given for the involved matrix, by which a one-dimensional vector is obtained from the original two-dimensional matrix. The obtained vector retains the computing ability and is used as the substitute for subsequent verification-related work. The work is increased in the domain of integer, while the work is decreased in the domain of group. Thus, in the algorithms **KeyGen** and **Compute** of the publicly verifiable scheme, the time complexity of verification-related work reduces dramatically from $O(n^2)$ to $O(n)$.

We summarize our main contributions as follows.

- We propose an efficient and publicly verifiable scheme MD-$\mathcal{VC}_{Matrix}$ for outsourced matrix multiplication.
- We propose to construct matrix digest for the original matrix, which is used as the substitute for the verification-related work in each algorithm of the scheme.
- We prove the security of the proposed scheme and demonstrate the efficiency by both theoretic analysis and experimental results simultaneously.

The rest of the paper is organized as follows. We give the system model, the security model and some definitions in Section 2. In Section 3, a publicly verifiable scheme is proposed for outsourced matrix multiplication and a fast algorithm is also presented for batch of exponentiations. We give the performance analysis in Section 4, including both the theoretic analysis and the experimental results. The related works are summarized in Section 5. Finally, we give the conclusions in Section 6.

2 Preliminaries

2.1 System Model and Security Model

We now give the system model adopted in this paper in Figure 1.

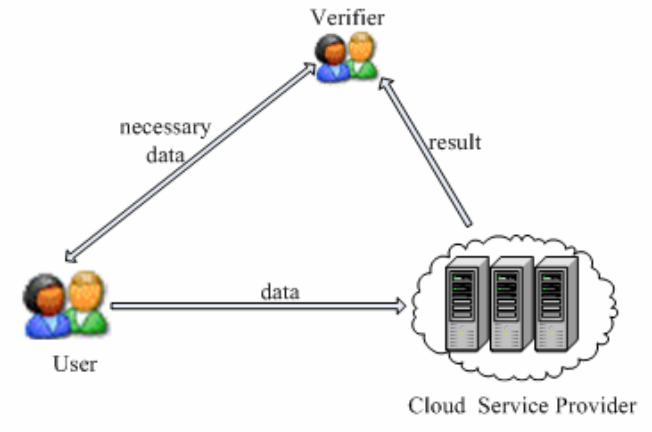


Figure 1. System model

Three parties are involved in the above model, which includes the cloud service provider (CSP), the user and the verifier. The CSP is deployed with abundant resources, and usually is a specific cloud service provider. The user owns a large-scale matrix M and a vector $\vec{x}$, and needs to compute $M\cdot\vec{x}$. The user is deployed with constrained resources and has no ability to perform the computing locally, and outsources the computing to the CSP. The verifier is an impartial third party, which computes with the publicly released data to determine whether or not the result is correct. The process of the outsourced computation is summarized as follows:

- The user performs computation on $M,\vec{x}$ respectively, to obtain the publicly released data for later verification, and sends part of the data to the verifier.
- The user sends $M,\vec{x}$ and the related data to the CSP.
- The CSP performs computation to get the result $M\cdot\vec{x}$ and the verification object, which are sent to the verifier.
- The verifier performs computation with the publicly released data the verification object to determine whether or not the result $M\cdot\vec{x}$ is correct.

We assume in this paper that the user and the verifier are both honest, and the CSP is semi-honest in the given model. We mainly study the security issue of public verification of the outsourced computation result, and privacy preservation issue is not taken into consideration in this paper.

2.2 Definitions

Definition 1. A publicly verifiable scheme $\mathcal{VC}$ for outsourced matrix multiplication is a four-tuple (**KeyGen**, **ProbGen**, **Compute**, **Verify**) [1, 4].

- **KeyGen** $(1^{\lambda},M)\rightarrow(SK_M,EK_M,PK_M)$: The user runs the algorithm with the security parameter λ on the input matrix M to get the keys SK_M,PK_M,EK_M, where SK_M is the secret key, PK_M is the public key, and EK_M is the evaluation key.

- **ProbGen** ($SK_M, \vec{x}) \to VK_x$: The user runs the algorithm with SK_M on the input vector $\vec{x}$ to get VK_x.
- **Compute** ($EK_M, \vec{x}) \to (\vec{y}, v)$: The CSP runs the algorithm to perform the matrix multiplication and obtains $\vec{y}, v$ as output, where $\vec{y}$ is the computation result of matrix multiplication and v is the verification object. The CSP can prove with v that he has performed the computing normally.
- **Verify** $(PK_M, VK_x, \vec{y}, v) \to true / false$: The third party verifier runs the algorithm to judge whether or not the result $\vec{y}$ is correct with PK_M, VK_x, v.

Let G_1, G_2, G_T be multiplicative cyclic groups of the same order q , and g_1, g_2 be a generator of group G_1, G_2 , respectively, where q be a big prime.

Definition 2. Unsymmetrical Bilinear Pairing.

A bilinear pairing $e: G_1 \times G_2 \to G_T$ defined on G_1, G_2, G_T is such a pairing that should satisfy the conditions as follows:

- bilinear. $\forall a, b \in \mathbb{Z}_q$, equation $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds.
- non-degenerate. The operations in equation $e(g, h) = 1$ holds, $g = 1$.
- computable. The operations in groups G_1, G_2, G_T , and operations of bilinear map e are solvable in probability polynomial time.

Definition 3. co-Computational Diffie-Hellman problem (co-CDH).

The advantage of solving the co-CDH by an adversary $\mathcal{A}$ is defined as

$$\mathcal{ADV}_{\mathcal{A}}^{cdh}(\lambda) = \Pr[(q, g_1, g_2, g_1^a, g_2^b) = g_1^{ab}],$$

where $a, b \in \mathbb{Z}_q$. Then we say the co-CDH assumption ϵ -holds in G_1, G_2 , if for every probability polynomial time (PPT) algorithm $\mathcal{A}$ we have $\mathcal{ADV}_{\mathcal{A}}^{cdh} \le \varepsilon$.

Definition 4. External Diffie-Hellman problem (XDH).

The advantage of $\mathcal{ADV}_{\mathcal{A}}^{cdh}(\lambda)$ of deciding the XDH problem by an adversary is defined as

$$\mathcal{ADV}_{\mathcal{A}}^{cdh}(\lambda) = |\Pr[(q, g_1, g_2, g_1^a, g_2^b, g_1^{ab})] - \Pr[(q, g_1, g_2, g_1^a, g_2^b, g_1^c)]| \le \varepsilon$$

where $a, b, c \in \mathbb{Z}_q$.

We say that the XDH assumption ε -holds over G_1, G_2, G_T , if for every PPT algorithm $\mathcal{A}$ we have $\mathcal{ADV}_{\mathcal{A}}^{xdh} \le \varepsilon$.

For any verifiable computation scheme $\mathcal{VC}$, we follow Fiore et al. to define the following experiment [1].

Definition 5. Experiment $Exp_{\mathcal{A}}^{PubVer}[\mathcal{VC}, f, \lambda]$.

$$(SK_f, EK_f, PK_f) \leftarrow KeyGen(1^{\lambda}, f)$$

For $i = 1$ to q

$$x_i \leftarrow \mathcal{A}(EK_f, \hat{x}_1, VK_{x,1}, \ldots, \hat{x}_{i-1}, PK_{x,i-1}, VK_{x,i-1})$$

$$(\hat{x}_i, PK_{x,i}, VK_{x,i}) \leftarrow ProbGen(f_\lambda, PK_f, E, \hat{M}, SK_f, x_i)$$

$$x^* \leftarrow \mathcal{A}(EK_f, \hat{x}_1, PK_{x,1}, VK_{x,1}, \ldots, \hat{x}_q, PK_{x,q}, VK_{x,q})$$

$$(\hat{x}^*, PK_{\hat{x}^*}, VK_{\hat{x}^*}) \leftarrow ProbGen(f_\lambda, PK_f, E, \hat{E}, SK_f, \hat{x}^*)$$

$$Res' \leftarrow \mathcal{A}(EK_f, \hat{x}_1, PK_{x,1}, VK_{x,1}, \ldots, \hat{x}_q, PK_{x,q}, VK_{x,q}, \hat{x}^*, PK_{\hat{x}^*}, SK_{\hat{x}^*})$$

$$\hat{y}' = Verify(PK_{\hat{x}^*}, VK_{\hat{x}^*}, Res')$$

$$y' \leftarrow Decryption(\hat{y}', H(\hat{E}'))$$

IF $\hat{y}' \neq \perp$ and $y' \neq f(x^*)$ output 1, otherwise output 0.

The variables x, y are both matrixes in our scheme, and x is in the domain of function f (where $f = M \cdot \vec{x}$ is a function that will be outsourced to the server).

For any $\lambda \in \mathcal{N}$, we define the advantage of an adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries in the above experiment against $\mathcal{VC}$ as:

$$Adv_{\mathcal{A}}^{PubVer}[\mathcal{VC}, f, q, \lambda] = \Pr[Exp_{\mathcal{A}}^{PubVer}[\mathcal{VC}, f, \lambda] = 1].$$

We say that a publicly verifiable computation scheme $\mathcal{VC}$ is secure for $\mathcal{F}$ if it holds that $Adv_{\mathcal{A}}^{PubVer}[\mathcal{VC}, f, q, \lambda]$ is negligible for any $f \in \mathcal{F}$ and any PPT adversary $\mathcal{A}$.

3 Our Proposed Scheme: MD-$\mathcal{VC}_{Matrix}$

We focus on an important security issue in this paper, which is the result public verification of the outsourced matrix multiplication. As far as the security issue of privacy preservation is concerned, an alternative would be the method adopted in [5-6], where the matrix M can be hidden by multiplying a sparse matrix P to get PM , which is sent to the server for later computing. In this section, we first give the definition of Matrix Digest. We then propose a scheme $MD - \mathcal{VC}_{Matrix}$ by using the matrix digest for public verification of outsourced matrix multiplication. Finally, a fast algorithm for computing batch of exponentiations is presented.

3.1 Matrix Digest

Definition 6. Matrix Digest (MD) is obtained by multiplying a matrix with a chosen vector.

Given a vector $\vec{p} = (p_1, \ldots, p_n) \in \mathbb{Z}^n$ and a matrix A ,

the MD of A is computed as $\vec{m} = \vec{p} \cdot A \in \mathbb{Z}^d$. The matrix A can be rewritten as $A = (\vec{a}_1, \ldots, \vec{a}_d)$, where $\vec{a}_i \in \mathbb{Z}^n (i = 1, \ldots, d)$ is a column vector. Thus, $\vec{m}_i = \vec{p} \cdot \vec{a}_i$, where $\cdot$ means inner product of vectors. MD has the following properties:

- **deterministic**. Given a vector $\vec{p}$, the matrix digest of a matrix is determined, that is, we can obtain different matrix digest for a matrix with different $\vec{p}$.
- **computable**. The matrix digest is essentially a vector and has the properties of vector, which can be applied to the operations of vector.
- **irreversible**. The computation of matrix digest is a one-way mapping. Given only a matrix digest, the matrix and the parameters cannot be detected. Even if the matrix digest and the parameters are given simultaneously, the matrix cannot be detected, too.

3.2 Scheme Details

We use the matrix digest instead of the original matrix for the verification-related computation. The original two-dimensional matrix is transformed into a one-dimensional vector by the matrix digest, and the computation of exponentiation is then decreased dramatically from $O(n^2)$ to $O(n)$.

The user outsources the computing of $\vec{y} = M \cdot \vec{x}$ to the CSP for better performance ratio, where $M \in \mathbb{Z}_p^{n \times d}$ is a matrix and $\vec{x} \in \mathbb{Z}_p^d$ is a vector, p is a large prime, and $n \geq 1, d \geq 1$ are integers. The third-party verifier can check with the publicly released information whether or not the returned result $\vec{y}$ is correct.

We now give the details of MD-$\mathcal{VC}_{Matrix}$ as follows.

KeyGen $(1^\lambda, M)$.

We assume that the input $M \in \mathbb{Z}_p^{n \times d}$ be a matrix. With the give security parameter λ, the user generates a description of bilinear groups $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$, a random vector $\vec{p} \in \mathbb{Z}_p^n$, a random integer $\alpha \in \mathbb{Z}_p$, and a key $K = (k_0, k_1, \ldots, k_d)$, where $k_i \in \mathbb{Z}_p$ $(i = 0, \ldots, d)$.

The user computes $\vec{m}, PK_p, \vec{w}$ as follows:

(1) $\vec{m} = \vec{p} \cdot M$, where $\cdot$ denotes here the multiplication of a vector and a matrix.

(2) $PK_p = (PK_1, \ldots, PK_n)$, where $PK_i = e(g_1^{\alpha p_i}, g_2)$.

(3) $\vec{w} = (w_1, \ldots, w_d)$, where $w_i = g_1^{\alpha m_i} \cdot F_K(i)$, and the symbol $\cdot$ denotes the multiplication of group elements.

Here, $F_K(i)$ is defined as an algebraic pseudorandom function, which is computed as follows:

$$F_K(i) = g_1^{k_0 k_i}, \ i = 1, \ldots, d.$$

Output (SK_M, EK_M, PK_M), where $SK_M = K$, $EK_M = (M, \vec{w})$, and $PK_M = PK_p$.

ProbGen $(SK_M, \vec{x})$.

Let a vector $\vec{x} = (x_1, ..., x_d) \in \mathbb{Z}_p^d$ be the input. The user computes the value $\rho_x = \prod_{i=1}^{d} F_K(i)^{x_i}$, and defines the verification key $VK_x = e(\rho_x, g_2)$. Output $(\vec{x}, VK_x)$.

Compute $(EK_M, \vec{x})$.

The CSP computes the result $\vec{y}$ and the verification object v as evidence, where $\vec{y} = M \cdot \vec{x}$ and $v = \prod_{i=1}^{d} w_i^{x_i}$. Output $(\vec{y}, v)$.

Verify $(PK_M, VK_x, \vec{y}, v)$.

The verifier checks with the public key and the verification key if

$$e(v, g_2) = \prod_{i=1}^{n} (PK_i)^{y_i} \cdot VK_x.$$

If the above equation holds then the algorithm outputs $\vec{y}$, otherwise outputs $\perp$.

Theorem 1. The algorithm **Verify** in our proposed scheme MD-$\mathcal{VC}_{Matrix}$ is correct.

Proof. By **KeyGen** and **Compute**, we have $\vec{m} = \vec{p} \cdot M$ and $\vec{y} = M \cdot \vec{x}$, respectively.

Then, $\vec{m} \cdot \vec{x} = \vec{p} \cdot \vec{y}$ can be obtained. Thus,

$$\sum_{i=1}^{d} m_i x_i = \sum_{i=1}^{n} p_i y_i.$$

Then, in **Verify**, when the third party verifier checks whether the equation $e(v, g_2) = \prod_{i=1}^{d} (PK_i)^{y_i} \cdot VK_x$ holds, we have

$$\begin{aligned} e(v, g_2) &= e(\prod_{i=1}^{d} w_i^{x_i}, g_2) \\ &= e(\prod_{i=1}^{d} g_1^{\alpha m_i x_i}, g_2) \cdot e(\prod_{i=1}^{d} F_K(i)^{x_i}, g_2) \\ &= e(g_1^{\sum_{i=1}^{d} \alpha m_i x_i}, g_2) \cdot VK_x \\ &= \prod_{i=1}^{n} e(g_1^{\alpha p_i y_i}), g_2) \cdot VK_x \\ &= \prod_{i=1}^{n} (PK_i)^{y_i} \cdot VK_x \end{aligned}$$

Thus, we conclude that the algorithm **Verify** works correctly.

3.3 Fast Algorithm

We observe that the running efficiency of the algorithm **KeyGen** can be improved further owning to the usage of so much exponentiation. The values $g_1^{\alpha p_i}, g_1^{\alpha m_i}$ are computed in **KeyGen** for later use, among which abundant duplicate computations are involved. As g_1 is a large number, the computations of $g_1^{\alpha p_i}, g_1^{\alpha m_i}$ are computationally expensive. We propose to compute the modular exponentiation incrementally. Thus, the duplicate computations are decreased and the running efficiency is improved further.

We give an algorithm **BatchExp** for efficient computation of batch of exponentiations. In the algorithm, the subroutine *Exp* is just a function to compute the exponentiation in common use, which returns $g^x \bmod N$ for invoking $Exp(g,x,N)$. We first sort the array $\vec{d}$ in ascending order. Then the result is obtained incrementally. Though the operation of sorting data is increased, the operation on computing exponentiation is decreased. The overall efficiency of computing batch of exponentiations is promoted.

Algorithm 1. BatchExp($\vec{d}, g, N$)

\ $sort(\vec{d})$ //in ascending order
\ $\vec{r} = (r_1, \ldots, r_n)$ //the computing result
$r_1 = Exp(g, d_1, N)$
\ FOR $i = 2$ to n
$r_i = r_{i-1} * Exp(g, d_i - d_{i-1}, N) mod N$
ENDFOR
return $\vec{r}$

3.4 Security Analysis

Following the work in [1-2], we give the security analysis of our scheme MD-$\mathcal{VC}_{Matrix}$ in this section.

Theorem 2. If $\mathcal{G}$ is such that co-CDH assumption ε_{cdh} -holds, and F is ε_{prf} -secure, then any PPT adversary $\mathcal{A}$ making at most $q = poly(\lambda)$ queries has advantage $ADV_{\mathcal{A}}^{PubVer}[MD - \mathcal{VC}_{Matrix}, f, q, \lambda] \le \varepsilon_{cdh} + \varepsilon_{prf}$.

We then define the following games, where $G_i(\mathcal{A})$ is the output of Game i run by the adversary $\mathcal{A}$.

Game 0. this game is the same as $Exp_{\mathcal{A}}^{PubVer}[MD - \mathcal{VC}_{Matrix}, f, q, \lambda]$.

Game 1. this game is similar to Game 0, except that $PK_M = (e(g_1^{ap_1}, g_2), \ldots, e(g_1^{ap_n}, g_2))$ in the **KeyGen** algorithm.

Game 2. this game is similar to Game 1, except that $\vec{w} = (w_1, \ldots, w_d)$, where $w_i = g_1^{am_i} \cdot R_i$.

The proof of this theorem is based on Games defined above, and is obtained by proving the following claims.

Claim 1 $\Pr[G_0(\mathcal{A}) = 1] = \Pr[G_1(\mathcal{A}) = 1]$

Proof. The difference between Game 0 and Game 1 is computing the public keys in **ProbGen**. However, due to the security of the bilinear map, the probability of the adversary winning in Game 1, i.e., $\Pr[G_1(\mathcal{A}) = 1]$, remains the same.

Claim 2 $|\Pr[G_1(\mathcal{A}) = 1] - \Pr[G_2(\mathcal{A}) = 1]| \le \varepsilon_{prf}$

Proof. The difference between Game 1 and Game 2 lies that each pseudo random function F_K is replaced with a random value of group R_i. Obviously, for any adversary $\mathcal{A}$, the difference between the possibility of winning two Games accounts on the possibility of winning the pseudo random function. Thus, the possibility of winning Game must be lower than winning a pseudo random function.

Claim 3 $\Pr[G_2(\mathcal{A}) = 1] \le \varepsilon_{cdh}$

Proof. Assume by contradiction that there exists a PPT adversary $\mathcal{A}$ such that the probability of $\mathcal{A}$ winning in Game 2 is a non-negligible function ε. Then, we show that we can build an efficient algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the co-CDH problem with probability $\varepsilon_{cdh} \ge \varepsilon$.

$\mathcal{B}$ takes as input $(q, g_1, g_2, G_1, G_2, G_T, e)$ and two random elements g_1^a, g_2^b, and proceeds with the following steps.

First, $\mathcal{B}$ chooses at random:

(1) a vector $\vec{p} = (p_1, \ldots, p_n) \in \mathbb{Z}_p^n$;

(2) another vector $\vec{w} = (w_i)_d \leftarrow G_1$ for $i = 1, \ldots, d$.

$\mathcal{B}$ then sets $EK_M = (M, \vec{w})$. Let $PK_M = (e(g_1^a, g_2^b)^{p_1}, \ldots, e(g_1^a, g_2^b)^{p_n})$. Then, $e(F_K(i), g_2) = e(w_i, g_2) / e(g_1^a, g_2^b)^{m_i}$.

The values PK_M, EK_M and $e(F_K(i), g_2)$ are distributed obviously as same as in Game 2.

Next, $\mathcal{B}$ runs $\mathcal{A}(PK_M, EK_M, e(F_K(i), g_2))$ and answers its queries as follows. Let $\vec{x}$ be the queried value, then $\mathcal{B}$ computes

$$VK_x = e(\prod_{i=1}^{d} F_K(i)^{x_i}, g_2) = \prod_{i=1}^{d} (e(w_i, g_2) / e(g_1^a, g_2^b)^{m_i})^{x_i}$$

and returns it to $\mathcal{A}$. By the bilinear property of $e(\cdot,\cdot)$, computing VK_x is equivalent to that in Game 2.

Finally, let $\hat{\sigma}_y = (\hat{y}, \hat{v})$ be the output of $\mathcal{A}$ at the end of the game, such that for some $\vec{x}^*$ chosen by $\mathcal{A}$, it holds that $Verify(PK_M, VK_{x^*}, \hat{\sigma}_y) = \hat{y}$, $\hat{y} \ne \perp$ and $y \ne M \cdot \vec{x}^*$. By verification, this means that

$$e(\hat{v}, g_2) = \prod_{i=1}^{n} PK_i^{\hat{y}_i} \cdot VK_{x^*} = \prod_{i=1}^{n} e(g_1^a, g_2^b)^{p_i \hat{y}_i} \cdot VK_{x^*} \quad \textbf{(1)}$$

Let $y = M \cdot \vec{x}^*$ be the correct computation result. Then, by correctness the following equation also holds:

$$e(v, g_2) = \prod_{i=1}^{n} e(g_1^a, g_2^b)^{p_i y_i} \cdot VK_{x^*} \quad \textbf{(2)}$$

So, dividing the equation (1) by (2), we have that

$$e(\hat{v}/v, g_2) = \prod_{i=1}^{n} e(g_1^{ab}, g_2)^{\sum_{i=1}^{d} p_i(\hat{y}_i - y_i)}$$

Because $\hat{y}_i \neq y_i$ ($i = 1, \ldots, n$), the value $\sum_{i=1}^{d} p_i(\hat{y}_i - y_i)$ is generally equal to 0. Thus,

$$g_1^{ab} = (\hat{v}/v)^{(\sum_{i=1}^{d} p_i(\hat{y}_i - y_i))^{-1}} \quad mod\ q$$

Therefore, if $\mathcal{A}$ wins in Game 2 with probability ε_{cdh}, then $\mathcal{B}$ solves co-CDH with the same probability. □

4 Performance Analysis

In this section, we discuss the performance of the scheme $\mathcal{VC}_{Matrix}$ [1] and our scheme MD-$\mathcal{VC}_{Matrix}$ by theoretic analysis and experimental results, respectively.

4.1 Theoretic Analysis

The symbol O is used traditionally to analyze an algorithm's time complexity. The computing in both $\mathbb{Z}$ and $\mathbb{G}$ is involved in our scheme MD-$\mathcal{VC}_{Matrix}$.

Thus, the time complexity analysis of MD-$\mathcal{VC}_{Matrix}$ cannot be done directly by O because of the difference of magnitude between the two fields. We utilize another two similar symbols $O_{\mathbb{Z}}, O_{\mathbb{G}}$, where $O_{\mathbb{Z}}$ denotes the time complexity of operations in $\mathbb{Z}$, and $O_{\mathbb{G}}$ denotes the time complexity of operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$.

Let the matrix $M \in \mathbb{Z}_p^{n \times d}$. We give the comparisons of computation overhead theoretically in Table 1 as follows.

Table 1. Comparison of computation overhead

Algorithm	$\mathcal{VC}_{Matrix}$	MD-$\mathcal{VC}_{Matrix}$
KeyGen	$O_{\mathbb{G}}(nd)$	$O_{\mathbb{Z}}(nd) + O_{\mathbb{G}}(d)$
ProbGen	$O_{\mathbb{G}}(n+d)$	$O_{\mathbb{G}}(d)$
Compute	$O_{\mathbb{Z}}(nd) + O_{\mathbb{G}}(nd)$	$O_{\mathbb{Z}}(nd) + O_{\mathbb{G}}(d)$

As can be seen from the Table 1, the efficiency of MD-$\mathcal{VC}_{Matrix}$ is superior to that of $\mathcal{VC}_{Matrix}$ in ProbGen and Compute. Regarding to the verification-related computation, the time cost of algorithm ProbGen is decreased from $O_{\mathbb{G}}(n+d)$ in $\mathcal{VC}_{Matrix}$ to $O_{\mathbb{G}}(d)$ in MD-$\mathcal{VC}_{Matrix}$, and the time cost of algorithm Compute is decreased $O_{\mathbb{G}}(nd)$ in $\mathcal{VC}_{Matrix}$ to $O_{\mathbb{G}}(d)$ in MD-$\mathcal{VC}_{Matrix}$.

It is the same in the algorithm Verify. As far as the algorithm KeyGen is concerned, it is difficult to differentiate MD-$\mathcal{VC}_{Matrix}$ from $\mathcal{VC}_{Matrix}$ with the symbols $O_{\mathbb{Z}}, O_{\mathbb{G}}$, because in the scheme MD-$\mathcal{VC}_{Matrix}$, the time cost is decreased in $O_{\mathbb{G}}$ and increased in $O_{\mathbb{Z}}$.

The difference can be further made clear only by experimental results.

4.2 Experiment Results

We now compare the running efficiency of the schemes with experiments. There are four algorithms in the publicly verifiable outsourced computation, i.e., KeyGen, ProbGen, Compute and Verify. We implement the first three algorithms of Fiore et al's scheme [1] and ours, respectively. As the computation overhead of algorithm Verify is the same according to the theoretic analysis in the two schemes, we will not discuss it here. We implement the algorithms with Java and the JPBC library [7]. The experiments are conducted on a laptop with an Intel Core i5 CPU running at 2.50GHz with 4GB RAMs. Each algorithm is executed 100 times, and the average time is gotten.

We first show the effect of the usage of the matrix digest. It is ambiguous to compare the algorithm **KeyGen** theoretically owning to the different magnitude in $\mathbb{Z}$ and $\mathbb{G}$. The computation overhead is $O(nd)$ to compute matrix digest in the scheme MD-$\mathcal{VC}_{Matrix}$, whereas it is done in $\mathbb{Z}$ with high efficiency.

The computations in $\mathbb{G}$ is computationally expensive, which includes bilinear map, modular exponentiation, etc. The comparison results between the scheme $\mathcal{VC}_{Matrix}$ in [1] and our proposed scheme MD-$\mathcal{VC}_{Matrix}$ are shown in Table 2, where for clarity, d is set to be equal to n.

Three sub-Tables are included in Table 2, which corresponds to the algorithm KeyGen, ProbGen and Compute, respectively. In each sub-table, t_1 is the time of the scheme $\mathcal{VC}_{Matrix}$, and t_2 is the time of our proposed scheme MD-$\mathcal{VC}_{Matrix}$. The value of the last column Speedup of each sub-table is gotten by t_2 / t_1.

Table 2. Effect of two schemes (Time in Milliseconds)

(a) Algorithm KeyGen

n	t_1	t_2	Speedup
100	44,805	3,713	12.07
300	401,185	11,068	36.25
500	1,120,142	18,378	60.95
700	2,195,619	25,683	85.49
900	3,624,704	33,055	109.66

(b) Algorithm ProbGen

n	t_1	t_2	Speedup
100	2,897	71	40.8
300	8,716	159	54.82
500	14,559	252	57.77
700	20,330	341	59.62
900	26,141	434	60.23

(c) Algorithm Compute

n	t_1	t_2	Speedup
100	4,425	45	98.33
300	39,743	133	298.82
500	114,906	229	501.77
700	219,196	316	693.66
900	365,458	406	900.14

As can be seen from the table, the efficiency of MD-$\mathcal{VC}_{Matrix}$ is superior to that of $\mathcal{VC}_{Matrix}$, which demonstrates the consistency with the theoretic analysis. Specifically, it is difficult to differentiate with theoretic analysis the algorithm KeyGen of two schemes. But it is easy to conclude with experiment results that the running efficiency is improved practically. For example, the algorithm KeyGen of our proposed scheme is almost 60 times faster than that of $\mathcal{VC}_{Matrix}$ when n is set to 500.

We then show the effect of the fast algorithm for batch of exponentiations proposed in Section 3.3, which is show in Table 3. In Table 3, the column $t_{original}$ is the time cost of a trivial method, where the batch of exponentiations is computed one by one with loop command. The column t_{fast} is the time cost of our proposed fast Algorithm. The value of the column of Speedup is gotten by $t_{original}/t_{fast}$. As can be seen that, the time cost of optimized algorithm is almost 25 percent of that of the original one.

Table 3. Effect of BatchExp (Time in Milliseconds)

n	$t_{original}$	t_{fast}	Speedup
10,000	4.4	0.87	5.06
20,000	8.86	1.82	4.87
30,000	13.3	2.83	4.70
40,000	17.75	3.86	4.60
50,000	22.24	4.95	4.49

5 Related Works

Abundant efforts have been done on securely outsourced computation, including linear system of equations [6, 11, 13], optimization problem [14-15], matrix operation [1-2, 5, 8-10], privacy preservation [23-25], data mining [26-27], secure multiparty computation [28-29], function query [30-31], etc. We focus on the security issue of correctness verification of the result, which can be grouped into two categories according to the executor of the result verification, i.e., private verification and public verification.

The private verification is performed by the user locally, which includes such methods as double server, redundant data, random check, Monte Carlo method, and so on. Hohenberger et al. used the double server method to verify the result of outsourced exponent operation [19]. Xie et al. proposed to insert fake tuples into the original dataset [16]. A portion of fake tuples should be included in the result. The result must be incorrect if the fake tuples that should be included in the result do not appear in the result. Hu et al. adopted the random check method to verify the result of outsourced matrix multiplication [5]. The user computes an item locally, if the item is not the same as that in the result, then the result is incorrect. Lei et al. applied the Monte Carlo method [9], which amounts to executing random check multiples times.

The public verification is performed by a third party verifier with public information provided by the user and server provider, which includes authentication data structure and computational hard problem. The verifier is acted by a neutral third party except the user and server provider. Devanbu et al. proposed to utilize Merkle Hash Tree (MHT) to verify the correctness of the query results on relational data [18]. Li et al. extended MHT to multi-dimensional data query and aggregate query [20], and Yang et al. further proposed MR-tree for k nearest neighbor (kNN) query on spatial data [21]. Hu et al. presented the voronoi neighbor techniques to separate the authentication structure with the query structure [22], by which the size of the verification object is diminished.

Fiore et al. utilized the discrete logarithm problem to propose a public verification scheme for outsourced matrix multiplication [1], where the property of Closed Form Efficiency is used to improve the running efficiency. Li et al. followed the framework to further improve the efficiency [2]. Elkhiyaoui pointed out in [17] that the secret key is used the algorithms of KeyGen and Compute on the user side in [1-2], which is unfavorable to public verification. A new verifiable computation scheme with completely public key on the user side is proposed in [16].

6 Conclusions

In this paper, we aim at efficiently and publicly verifiable computation of outsourced matrix multiplication. We adopt two strategies, one is the matrix digest, and the other is fast computation for batch of exponentiations. The verification-related computation is decreased by the matrix digest, which improves the running efficiency dramatically. The running efficiency of the algorithm KeyGen is also improved by the fast algorithm for batch of exponentiations. The performance analysis on theoretic analysis and experimental results jointly demonstrate the efficiency of our proposed scheme. In future work, we plan to present more efficient scheme for publicly verifiable computation of outsourced matrix multiplication.

Acknowledgments

This work was supported in part by the Foundation of National Natural Science of China (No. 61772147, 61772124), Guangdong Province Natural Science Foundation of major basic research and Cultivation project (No. 2015A030308016), Project of Ordinary University Innovation Team Construction of Guangdong Province (No. 2015KCXTD014), Basic Research Major Projects of Department of education of Guangdong Province (No. 2014KZDXM044), Collaborative Innovation Major Projects of Bureau of Education of Guangzhou City (No. 1201610005), Humanities and Social Science Project of Education (No. 15YJCZH111), and Fundamental Research Funds for the Central Universities (No. N150402002).

References

[1] D. Fiore, R. Gennaro, Publicly Verifiable Delegation of Large Polynomials and Matrix Computations, with Applications, *19th ACM Conference on Computer and Communications Security*, Raleigh, NC, 2012, pp. 501-512.

[2] H. Li, S. Zhang, T. H. Luan, H. Ren, Y. Dai, L. Zhou, Enabling Efficient Publicly Verifiable Outsourcing Computation for Matrix Multiplication, *IEEE International Telecommunication Networks and Applications Conference*, Sydney, Australia, 2015, pp. 44-50.

[3] K. Jia, H. Li, D. Liu, S. Yu, Enabling Efficient and Secure Outsourcing of Large Matrix Multiplications, *IEEE Global Communications Conference*, San Diego, CA, 2015, pp. 1-6.

[4] R. Gennaro, C. Gentry, B. Parno, Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers, in: T. Rabin (Ed.), *Advances in Cryptology – CRYPTO 2010*, *Lecture Notes in Computer Science*, Vol. 6223, Springer, Berlin, 2010, pp. 465-482,.

[5] X. Hu, D. Pei, C. Tang, D. Wong, Verifiable and Secure Outsourcing of Matrix Calculation and Its Application, *Scientia Sinica (Informationis)*, Vol. 43, No. 7, pp. 842-852, July, 2013.

[6] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, D. 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, January, 2015.

[7] A. D. Caro, V. Iovino, jpbc: Java Pairing based Cryptography, *16th IEEE Symposium on Computers and Communications*, Kerkyra, Greece, 2011, pp. 850-855.

[8] M. Atallah, K. Frikken, Securely Outsourcing Linear Algebra Computations, *5th ACM Symposium on Information, Computer and Communications Security*, Beijing, China, 2010, pp. 48-59.

[9] X. Lei, X. Liao, T. Huang, H. Li, C. Hu, Outsourcing Large Matrix Inversion Computation to A Public Cloud, *IEEE Transactions on Cloud Computing*, Vol. 1, No. 1, pp. 78-87, January-June, 2013.

[10] X. Lei, X. Liao, T. Huang, 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, October, 2014.

[11] C. Wang, K. Ren, J. Wang, Q. Wang, Harnessing the Cloud for Securely Outsourcing Large-scale Systems of Linear Equations, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 6, pp. 1172-1181, June, 2013.

[12] F. Chen, T. Xiang, Y. Yang, Privacy-preserving and Verifiable Protocols for Scientific Computation Outsourcing to the Cloud, *Journal of Parallel and Distributed Computing*, Vol. 74, No. 3, pp. 2141-2151, March, 2014.

[13] S. Salinas, C. Luo, X. Chen, P. Li, Efficient Secure Outsourcing of Large-scale Linear Systems of Equations, *IEEE International Conference on Computer Communications*, Hong Kong, China, 2015, pp. 1035-1043.

[14] F. Ma, G. Sheng, Y. Yin, A Superlinearly Convergent Method for the Generalized Complementarity Problem over a Polyhedral Cone, *Journal of Applied Mathematics*, Vol. 2013, Article ID 671402, pp. 1-6, July, 2013.

[15] C. Wang, K. Ren, J. Wang, Secure and Practical Outsourcing of Linear Programming in Cloud Computing, *IEEE International Conference on Computer Communications*, Shanghai, China, 2011, pp. 820-828.

[16] M. Xie, H. Wang, J. Yin, X. Meng, Integrity Auditing of Outsourced Data, *33rd International Conference on Very Large Data Bases*, Vienna, Austria, 2007, pp. 782-793.

[17] K. Elkhiyaoui, M. Onen, M. Azraoui, R. Molva, Efficient Techniques for Publicly Verifiable Delegation of Computation, *11th ACM on Asia Conference on Computer and Communications Security*, Xi'an, China, 2016, pp. 119-128.

[18] P. Devanbu, M. Gertz, C. Martel, S. G. Stubblebine, Authentic Data Publication over the Internet, *Journal of Computer Security*, Vol. 11, No. 3, pp. 291-314, July, 2003.

[19] S. Hohenberger, A. Lysyanskaya, How to Securely Outsource Cryptographic Computations, in: J. Kilian (Ed.), *Theory of*

Cryptography, TCC 2005, Lecture Notes in Computer Science, Vol. 3378, Springer, Berlin, pp. 264-282, 2005.

[20] F. Li, M. Hadjieleftheriou, G. Kollios, L. Reyzin, Authenticated Index Structures for Aggregation Queries, *ACM Transactions on Information and System Security*, Vol. 13, No. 4, Article No. 32, December, 2010.

[21] Y. Yang, S. Papadopoulos, D. Papadias, G. Kollios, Authenticated Indexing for Outsourced Spatial Databases, *The VLDB Journal*, Vol. 18, No. 3, pp. 631-648, June, 2009.

[22] L. Hu, W. S. Ku, S. Bakiras, C. Shahabi, Verifying Spatial Queries Using Voronoi Neighbors, *18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Jose, CA, 2010, pp. 350-359.

[23] Z. Fu, J. Shu, J. Wang, Y. Liu, S. Lee, Privacy-Preserving Smart Similarity Search Based on Simhash over Encrypted Data in Cloud Computing, *Journal of Internet Technology*, Vol. 16, No. 3, pp. 453-460, May, 2015.

[24] G. Sheng, T. Wen, Q. Guo, Y. Yin, Privacy Preserving Inner Product of Vectors in Cloud Computing, *International Journal of Distributed Sensor Networks*, Vol. 2014, pp. 1-6, May, 2014.

[25] Z. Wang, Z. Chu, Efficient Mediated Ciphertext-Policy Attribute-Based Encryption for Personal Health Records Systems, *Journal of Internet Technology*, Vol. 16, No. 5, pp. 877-883, September, 2015.

[26] W. Xue, H. Li, Y. Peng, J. Cui, Y. Shi, Secure k Nearest Neighbors Query for High-dimensional Vectors in Outsourced Environments, *IEEE Transactions on Big Data*, August, 2017.

[27] Y. Zhao, J. X. Yu, G. Wang, L. Chen, B. Wang, G. Yu, Maximal Subspace Coregulated Gene Clustering, *IEEE Transactions on Knowledge & Data Engineering*, Vol. 20, No. 1, pp. 83-98, January, 2008.

[28] Y. Sun, Q. Wen, Y. Zhang, H. Zhang, Z. Jin, W. Li, Two-cloud-servers-assisted Secure Outsourcing Multiparty Computation, *The Scientific World Journal*, Vol. 2014, pp. 1-7, May, 2014.

[29] C. Tang, S. Gao, C. Zhang, The Optimal Linear Secret Sharing Scheme for Any Given Access Structure, *Journal of Systems Science & Complexity*, Vol. 26, No. 4, pp. 634-649, August, 2013.

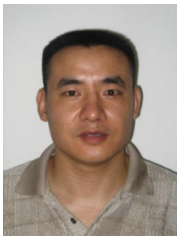
[30] G. Sheng, C. Tang, H. Han, W. Gao, X. Hu, Authentication of Outsourced Linear Function Query with Efficient Update, *Cluster Computing*, July, 2017.

[31] C. Xiang, C. Tang, New Verifiable Outsourced Computation Scheme for An Arbitrary Function, *International Journal of Grid and Utility Computing*, Vol. 7, No. 3, pp. 190-199, 2016.

Biographies

Gang Sheng is a post-doctoral in College of Mathematics and Information Science, Guangzhou University, China. He received his Ph.D. degree in computer application technology from Northeastern University, China in 2015. His research interests include cryptography and cloud computing. He is a member of Chinese Association for Cryptologic Research.

Chunming Tang is a professor in College of Mathematics and Information Science, Guangzhou University, China. He received the Ph.D. degree in applied mathematics from Chinese Academy of Sciences, China in 2004. His research interests include cryptography and secure multiparty computing. He is a member of Chinese Association for Cryptologic Research.



Wei Gao is an associate professor in School of Mathematics and Statistics Science, Ludong University, China. He received his Ph.D. degree in applied mathematics from Hunan University, China in 2006. His research interests include cryptography and number theory. He is a member of Chinese Association for Cryptologic Research.

Ying Yin is an associate professor in School of Computer Science and Engineering, Northeastern University, China. She received her Ph.D. degree in computer science from Northeastern University, China in 2008. Her major research interests include data mining and machine learning. She is a member of IEEE ACM and a member of CCF.

Yunlu Cai is an associate professor in College of Mathematics and Information Science, Guangzhou University, China. His research interests include cryptography, secure multiparty computing and outsourced computing.