RESEARCH **Open Access**

Publicly verifiable and efficiency/security-adjustable outsourcing scheme for solving large-scale modular system of linear equations

Panpan Meng[1], Chengliang Tian[1,2,3*] and Xiangguo Cheng[1]

Abstract

Solving large-scale modular system of linear equations ($\mathcal{LMSLE}$) is pervasive in modern computer and communication community, especially in the fields of coding theory and cryptography. However, it is computationally overloaded for lightweight devices arisen in quantity with the dawn of the things of internet (IoT) era. As an important form of cloud computing services, secure computation outsourcing has become a popular topic. In this paper, we design an efficient outsourcing scheme that enables the resource-constrained client to find a solution of the $\mathcal{LMSLE}$ with the assistance of a public cloud server. By utilizing affine transformation based on sparse unimodular matrices, our scheme has three merits compared with previous work: 1) Our scheme is efficiency/security-adjustable. Our encryption method is dynamic, and it can balance the security and efficiency to match different application scenarios by skillfully control the number of unimodular matrices. 2) Our scheme is versatile. It is suit for generic *m*-by-*n* coefficient matrix $\mathbf{A}$, no matter it is square or not. 3) Our scheme satisfies public verifiability and achieves the optimal verification probability. It enables any verifier which is not necessarily the client to verify the correctness of the results returned from the cloud server with probability 1. Finally, theoretical analysis and comprehensive experimental results confirm our scheme's security and high efficiency.

Keywords: Cloud computing, Secure computation outsourcing, Modular system of linear equations, Unimodular matrix transformation, Privacy-preserving

Introduction

With the rapid development of 5G technologies, the Internet of things (IoT) era is coming. More and more intelligent devices are connected to the internet and communicate with each other, which will greatly facilitate people's life [1–3]. It is predicted by the technology research firm Gartner that there will be 26 billion smart devices on the Internet of Things (IoT) by 2020 [4]. However, quantities of these devices, such as wearable devices, home appliances, and RFID cards, suffer from limited computing ability, low storage space and constrained communication bandwidth. It is unrealistic for these resource-constrained devices to perform large-scale data computation and storage task. Fortunately, cloud computing, as a type of Internet-based computing, offers availably abundant processing resources to electronic devices on demand. Under this promising computing paradigm, the resource-constrained clients can outsource their overloaded computations and storages to the resource-abundant cloud servers without the investment of purchasing and maintaining their own computing facilities. Although it brings many advantages, the fact that clients and cloud servers are not necessarily in the same trusted domain brings many security issues and challenges [5–7]. On the one hand, the clients' outsourcing data may contain their sensitive information such as proprietary research data, private asset records, and personal health information, etc. The exposure of these critical information could incur the severe loss of clients' credit, economic, spirit, life

*Correspondence: tianchengliang@qdu.edu.cn
[2]Business School, Qingdao University, Qingdao 266071, China
[3]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
Full list of author information is available at the end of the article

and asset. On the other hand, the physical isolation from the clients and external ill-disposed economic incentives may make cloud servers curious, lazy and even malicious, and then steal clients' private information and return random, or deliberately forged results to the clients. Therefore, a security outsourcing scheme should fulfill the following properties [8, 9]: (1) Correctness. The clients should obtain the correct computation results if the cloud servers conduct the assigned computation task honestly. (2) Input/output privacy. The curious or even malicious cloud servers can not steal or speculate clients' actual input/output information. (3) Verifiability. The clients can verify the correctness of the results returned from the cloud servers. (4) Efficiency. The scheme must be efficient. That is, the clients' time/space cost of performing the large-scale data computation/storage task should be substantially cheaper than that of performing the task by themselves.

For some given integer q, a vector $\mathbf{b} \in \mathbb{Z}_q^n$ and a full (column or row) rank matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, solving the modular systems of linear equations

$$\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q \tag{1}$$

is a fundamental computational problem in modern computer algebra [10]. It has various applications in information and communications fields. For example, in coding theory, we needs to solve this problem to decode q-ary linear code [11, 12], and, in lattice-based cryptography, it is used for generating signatures and private keys in the signature and encryption/decryption schemes based on Small Integer Solution (SIS) [13] and Learning with Errors (LWE) problems [14–16]. Also, it frequently emerges in machine learning [17]. The classic algorithm of solving this problem is Gaussian elimination with a time complexity of $O(mn^2(\log q)^2)$. It is efficient when the size of $\mathbf{A}$ is small. However, the exponential growth in the quantity of data generated in IoT era makes us often have to deal with large-scale matrices. For example, the recent post-quantum key exchange protocol [18] requires $m, n \geq 1024$ and $q = 2^{32} - 1$, and its improved variant [19] also requires $m, n \geq 1024$ and $q = 12289$. For the large-scale matrix $\mathbf{A}$, the Gaussian elimination becomes time-consuming which may result in the infeasible for the resource-constrained clients to solve this problem. Therefore, it is of great necessity to design an efficient algorithm to securely outsourcing the solving of the large-scale modular system of linear equations ($\mathcal{LMSLE}$).

Obviously, when $\mathbf{A}$ is square and invertible modulo q, the ($\mathcal{LMSLE}$) problem (1) has a unique solution in $\mathbb{Z}_q$ which can be denoted as $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \mod q$. In this case, we can realize the securely outsourcing of ($\mathcal{LMSLE}$) by designing an efficient method to outsource the inverse of $\mathbf{A}$ modulo q. Although many researches have considered the outsourcing of large-scale matrix inversion computation [20–22], they focused on matrices over the real field $\mathbb{R}$ or the finite field $\mathbb{F}_q$ and the probability of verifiability can not achieve the optimal 1. Furthermore, the existing methods only designed for a square and invertible matrix $\mathbf{A}$, while here we consider a general m-by-n matrix $\mathbf{A}$. Another viable way is directly considering the secure outsourcing of $\mathcal{LMSLE}$. Through imitating the currently most efficient outsourcing algorithm of large-scale system of linear equations ($\mathcal{LSLE}$) [23], one may want to blind the input ($\mathbf{A}, \mathbf{b}$) and the output $\mathbf{x}$ by sparse matrix transformations and translation vector. However, just as the authors' mentioned in the paper, their method can not provide strong enough privacy. Therefore, how to design an efficient and strong enough security outsourcing scheme for a general input matrix $\mathbf{A}$ is left as a meaningfully practical problem.

Our contributions

To address the above-mentioned problem, in this paper, we put forward an efficiency/security-adjustable outsourcing scheme to find a solution vector of the $\mathcal{LMSLE}$ problem $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$ in case that the problem is solvable. Also, our scheme is publicly variable, i.e. any verifier (not necessarily the client) can assess the correctness of the server's results. The main technique involved in our scheme essentially can be treated as an improvement of Zhang et al.'s technique [22] and Chen et al.'s technique [23], which makes our scheme superior in the following three aspects:

1) Our scheme provides an adjustable encryption method based on successively sparse unimodular matrix transformations. That is, we can adjust the number of sparse unimodular matrices in the scheme to balance security and efficiency to match the different application scenarios.
2) Our scheme is applicable to any matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$ is solvable, no matter it is square or non-square.
3) Our scheme are publicly verifiable and achieves the optimal verifiability probability 1. In other words, our scheme allows the client to delegate the verification task to any honest or even curious edge server without leaking the client's private information. Additionally, the verifier (e.g. the client or the third-party edge server) can detect the cloud server's misbehaviors with optimal probability 1.

Organization

The rest of our paper is arranged as follows: In "Security model and definitions" section, we illustrate the system model and related definitions in our outsourcing design of $\mathcal{LMSLE}$. "Preliminaries" section reviews some necessary preliminaries used in the design of our scheme. The main scheme and its correctness, privacy, verifiability and efficiency analysis are presented in "Main scheme" section, and "Performance evaluation" section evaluates the practical performance of the proposed scheme by comprehensive experiments. We survey the related work in "Related work" section. Finally, we conclude our paper in "Conclusion and future direction" section.

Security model and definitions

System model

As shown in Fig. 1, the system model of our secure outsourcing scheme $\mathcal{SOS}_{\mathcal{LMSLE}}$ $(q, \mathbf{A}, \mathbf{b})$ involves three entities: a resource-constrained client *C*, a remote public cloud server *S*, and an edge server *E* closed to *C*.

Client: The client *C* is with constrained computing resource and storage space. It intends to securely find a solution of the large-scale modular system of linear equations $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$ by leveraging the cloud server's computation resource. To keep the privacy of the input $(\mathbf{A}, \mathbf{b})$ and the output (solution vector) $\mathbf{x}$, *C* firstly generates a secret key *SK* and utilizes it to encrypt $(\mathbf{A}, \mathbf{b})$ into $(\mathbf{A}', \mathbf{b}')$, and then sends $(q, \mathbf{A}', \mathbf{b}')$ to *S* and *E*.

Cloud server: The cloud server *S* is public and resource-abundant, yet it is far from the client *C* and thus maybe untrusted. After receiving $(q, \mathbf{A}', \mathbf{b}')$, *S* performs the assigned computation task that solving another large-scale modular system of linear equations $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$, and returns the solution vector $\mathbf{x}'$ to *E*.

Edge server: The edge server *E* is semi-public and has more computing power and larger storage space than that of the client *C*. However, compared with the cloud server *S*, edge server's resource is also limited, so it cannot support complex computation task. In our outsourcing system, it mainly assists the client to verify the correctness of the returned result from *S*. Noteworthily, the verification task also can be conducted by the client itself, and thus

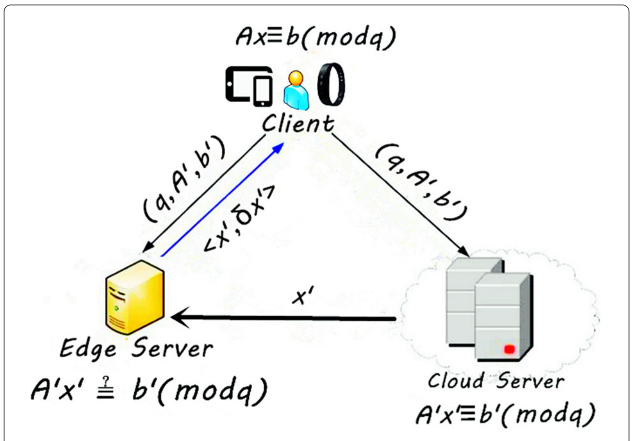

Fig. 1 System model

this entity is optional in our system and here mainly used to illustrate the public verifiability of our scheme.

Formally, the framework of our system model consists of the following five probabilistic polynomial time (PPT) algorithms:

1) **KeyGen**$(\Phi, 1^m, 1^n) \rightarrow \{SK\}$: On input a $\mathcal{LMSLE}$ problem instance $\Phi = (q, \mathbf{A}, \mathbf{b})$ with $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, this algorithm performed by the client *C* generates a random secret key *SK* which should be kept secret by *C*.
2) **CEnc**$(\Phi, SK) \rightarrow \{\Phi'\}$: Utilizing the secret *SK*, the client *C* performs this algorithm to encrypt the input instance $\Phi = (q, \mathbf{A}, \mathbf{b})$ into a blind input $\Phi' = (q, \mathbf{A}', \mathbf{b}')$, and sends Φ' to the cloud server *S* and the edge server *E*.
3) **SCom**$(\Phi') \rightarrow \{\mathbf{x}'\}$: After receiving the encrypted input $\Phi' = (q, \mathbf{A}', \mathbf{b}')$, the cloud server *S* invokes this algorithm to solve the large-scale modular system $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$, and returns a solution vector $\mathbf{x}'$ to the edge server *E*.
4) **EVer**$(\mathbf{x}', \Phi') \rightarrow \{\mathbf{x}', \delta_{\mathbf{x}'}\}$: This algorithm performed by the edge server *E* firstly verifies the correctness of the $\mathbf{x}'$ returned from *S*. If $\mathbf{x}'$ is correct, then $\delta_{\mathbf{x}'} = 1$. Else, $\delta_{\mathbf{x}'} = 0$. At last, the algorithm sends $(\mathbf{x}', \delta_{\mathbf{x}'})$ to the client *C*.
5) **CDec**$(\mathbf{x}', \delta_{\mathbf{x}'}, SK) \rightarrow \mathbf{x} \cup \perp$: This algorithm is performed by the client *C*. If the input $\delta_{\mathbf{x}'} = 1$, then the algorithm uses the secret key *SK* to decrypt $\mathbf{x}'$ into the actual solution vector $\mathbf{x}$. Otherwise, the algorithm outputs $\perp$.

Threat model

Standing at the client's perceptive, the threats in our computation outsourcing system mainly originate from the credibility of the cloud server *S* and the edge server *E*. According to the misbehaviors of servers, the threats mainly come from the following two types of servers: the *honest-but-curious (HBC)* server, and the *fully malicious (FM)* server.

HBC server: the server will perform the specified computation task honestly and return the correct calculation result. However, it is curious about the actual input/output information, and tries to steal the client's valuably private information for selfish purposes.

FM server: the server not only wants to steal the client's private information, but also may arbitrarily deviate from the specified computation task and returns a random or even a tactical forged result to fool the client.

Clearly, For the *HBC* server, a secure outsourcing scheme must ensure the privacy of the client's input/output information. While, For the *FM* server, an outsorucing scheme is secure if it simultaneously satisfies input/output privacy and the returned result's verifiability.

In the light of the different misbehaviors of the two servers in our system model, there exist four possible threat models: *HBC*-cloud + *HBC*-edge, *FM*-cloud + *HBC*-edge,*HBC*-cloud + *FM*-edge, and *FM*-cloud+*FM*-edge. Since the cloud server is remote and out of control, and the edge server is semi-public which can be deemed as an interior small server for a group, we consider the threat model as the combination of a *FM* cloud server and a *HBC* edge server.

Design goals

To design an efficient outsourcing scheme under the *FM*-cloud + *HBC*-edge servers model, the scheme should at least fulfill four properties: *correctness, privacy, verifiability* and *efficiency*.

The first requirement *correctness* means that the scheme enables the client to acquire one solution vector of the $\mathcal{LMSLE}$ problem if the cloud server honestly performs the specified computation task.

Definition 1 (**Correctness**) *A secure outsourcing computation scheme* $\mathcal{SOS}_{\mathcal{LMSLE}}(\cdot)$ *is correct if, for any valid input* $\Phi = (q, \mathbf{A}, \mathbf{b})$, *the key generation algorithm produces* $\{SK\} \leftarrow \mathbf{KeyGen}(\Phi, 1^m, 1^n)$ *such that, if* $\{\Phi' = (q, \mathbf{A}', \mathbf{b}')\} \leftarrow \mathbf{CEnc}(\Phi, SK)$, $\mathbf{x}' \leftarrow \mathbf{SCom}(\Phi')$ *and* $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$, *the algorithm* $\mathbf{CDec}(\mathbf{x}', \delta_{\mathbf{x}'}, SK)$ *outputs* $\mathbf{x}$ *subject to* $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$.

The second requirement is *privacy* which means the scheme should guarantee a *FM* or a *HBC* server can not obtain the client's actual input/output information with a overwhelming probability.

Definition 2 (**Input/Output privacy**) *A secure outsourcing computation scheme* $\mathcal{SOS}_{\mathcal{LMSLE}}(\cdot)$ *satisfies the input (resp. output) privacy if, for any valid input* $\Phi = (q, \mathbf{A}, \mathbf{b})$, *the key generation algorithm produces* $\{SK\} \leftarrow \mathbf{KeyGen}(\Phi, 1^m, 1^n)$ *such that the probability that the cloud/edge server can recover* $(\mathbf{A}, \mathbf{b})$ *(resp.* $\mathbf{x}$ *satisfying* $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$*) is negligible even if the cloud/edge server knows* $\{\Phi' = (q, \mathbf{A}', \mathbf{b}')\} \leftarrow \mathbf{CEnc}(\Phi, SK)$ *and* $\mathbf{x}' \leftarrow \mathbf{SCom}(\Phi')$.

The third requirement *verifiability* means the scheme can verify the correctness of the result returned from the cloud server with a nonnegligible probability.

Definition 3 (**$(1-\beta)$-Verifiability**) *A secure outsourcing computation scheme* $\mathcal{SOS}_{\mathcal{LMSLE}}(\cdot)$ *is* $(1-\beta)$*-verifiability if, for any valid input* $\Phi = (q, \mathbf{A}, \mathbf{b})$, *the key generation algorithm produces* $\{SK\} \leftarrow \mathbf{KeyGen}(\Phi, 1^m, 1^n)$ *such that, if* $\{\Phi' = (q, \mathbf{A}', \mathbf{b}')\} \leftarrow \mathbf{CEnc}(\Phi, SK)$, *and* $\{\mathbf{x}'\} \leftarrow \mathbf{SCom}(\Phi')$, *the probability of* $\mathbf{EVer}(\mathbf{x}', \Phi')$ *outputting* $\delta_{\mathbf{x}'}$ *satisfies*

$$\Pr[\,\delta_{\mathbf{x}'} = 1 \leftarrow \mathbf{EVer}(\mathbf{x}', \Phi') \mid \mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q] = 1,$$
$$\Pr[\,\delta_{\mathbf{x}'} = 1 \leftarrow \mathbf{EVer}(\mathbf{x}', \Phi') \mid \mathbf{A}'\mathbf{x}' \neq \mathbf{b}' \mod q] \leq \beta.$$

Finally, the scheme should be efficient. In other words, the scheme should enable the client to achieve substantial computation savings compared to performing the work on client's own.

Definition 4 (α-**Efficiency**) *A secure outsourcing computation scheme* $\mathcal{SOS}_{\mathcal{LMSLE}}(\cdot)$ *is* α*-efficiency if, suppose,for any valid input* $\Phi = (q, \mathbf{A}, \mathbf{b})$, *the client's time overhead of solving* $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$ *on its own is* $t_{original}$, *and the local-client's time overhead of performing the task by performing the outsourcing algorithm* $\mathcal{SOS}_{\mathcal{LMSLE}}(\Phi)$ *is* t_{client}, $\frac{t_{original}}{t_{client}} \geq \alpha$.

It is worth mentioning that the factor α in the above definition measures the level of outsourcing scheme's efficiency. Clearly, The larger the factor α becomes, the more computational savings the local-client achieves and the more efficient the scheme $\mathcal{SOS}_{\mathcal{LMSLE}}(\cdot)$ is.

Preliminaries

In this section, we review the frequently used symbols in this paper and some necessary basic concepts involved the design of our scheme.

Notations and terminologies

Throughout the paper, we use capital and bold (resp. lower-case and bold) letters to denote matrices (resp. vectors). For some matrix $\mathbf{A}$ (resp. vector $\mathbf{b}$), $\mathbf{A}^{\mathrm{T}}$ (resp. $\mathbf{b}^{\mathrm{T}}$) denotes the transposition of matrix $\mathbf{A}$ (resp. vector $\mathbf{b}$), $\det(\mathbf{A})$ denotes the determinant of $\mathbf{A}$. Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}_q = \{0, 1, \cdots, q-1\}$ denote the residue ring $\mathbb{Z}/q\mathbb{Z}$. $\mathbb{Z}_q^{m\times n}$ represents the set of all the $m \times n$ matrices whose entries belong to $\mathbb{Z}_q$.

Unimodular matrix

Unimodular matrices are a special kind of integer matrices, which has wide applications in computer science community, especially in matrix computation theory, coding theory and lattice-based cryptography [24].

Definition 5 (Unimodular matrix [25]) *An n-by-n integer matrix* $\mathbf{U}$ *is unimodular if and only its determinant* $\det(\mathbf{U}) = 1$ *or* $\det(\mathbf{U}) = -1$.

A very "nice" property of unimodular matrix is that the inverse of an unimodular matrix is still unimodular. Here we list it as a lemma and omit its proof.

Lemma 1 ([26]) *For some unimodular matrix* $\mathbf{U} \in \mathbb{Z}^{n\times n}$, *there exists a unique matrix* $\mathbf{V} \in \mathbb{Z}^{n\times n}$ *such that*

$\mathbf{UV} = \mathbf{I}_{n\times n}$ *and* $|\det(\mathbf{V})| = 1$, *where* $\mathbf{I}_{n\times n}$ *denotes the n-by-n identity matrix.*

For instance, for the unimodular matrix $\mathbf{U} = \begin{pmatrix} 3 & 2 \\ 4 & 3 \end{pmatrix}$, its inverse is $\mathbf{V} = \begin{pmatrix} 3 & -2 \\ -4 & 3 \end{pmatrix}$ with $\det(\mathbf{V}) = 3 \times 3 - (-2) \times (-4) = 1$, which is trivially unimodular.

Now, we prove another important result about the unimodular matrix, which is very useful in the forthcoming analysis of our scheme's input/output privacy in "Input/output privacy" section.

Lemma 2 *Let* $\mathcal{P}$ *denote the set*

$$\left\{\mathbf{X} \mid \mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \in \mathcal{X}^{2\times 2} \text{ and } \det(\mathbf{V}) = \pm 1\right\}, \tag{2}$$

where $\mathcal{X} = \mathbb{Z} \cap (-2^\lambda, 2^\lambda)$ *denotes the set of the integers with bit length no larger than* λ. *Then the size of* $\mathcal{P}$ *satisfies*

$$\#\mathcal{P} \geq \frac{48}{\pi^2}(2^\lambda - 1)^2.$$

Proof From the definition of $\mathcal{P}$, $\forall \mathbf{X} \in \mathcal{T}$, $|\det(\mathbf{X})| = |x_{11}x_{22} - x_{21}x_{12}| = 1$. Since the number of the unimodular matrices with determinant 1 is the same as that of the unimodular matrices with determinant -1, we only count the number of unimodular matrices with determinant 1 (i.e. $x_{11}x_{22} - x_{21}x_{12} = 1$).

Since $x_{11}x_{22} - x_{21}x_{12} = 1$, we have $\gcd(x_{11}, x_{21}) = 1$. Clearly, for any relatively prime pair $(a, b) \in \mathcal{X} \times \mathcal{X}$, there exists at least one unimodular matrix in $\mathcal{P}$. According to the proof of Theorem 3.9 in reference [27], the number of relatively prime pairs in $\mathcal{X} \times \mathcal{X}$ is no less than $\frac{24}{\pi^2}(2^\lambda - 1)^2$. Therefore, the number of unimodular matrices with determinant 1 is at least $\frac{24}{\pi^2}(2^\lambda - 1)^2$.

Consequently, the size of $\mathcal{P}$ satisfies

$$\#\mathcal{P} \geq 2\frac{24}{\pi^2}(2^\lambda - 1)^2 = \frac{48}{\pi^2}(2^\lambda - 1)^2.$$

□

Main scheme

A bird's-eye view of the Main idea

Given an integer q, a vector $\mathbf{b} \in \mathbb{Z}_q^m$ and a large-scale full (column or row) rank matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, we intend to find a solution vector $\mathbf{x}$ such that

$$\mathbf{Ax} \equiv \mathbf{b} \mod q \tag{3}$$

in case that the system is solvable. Since the size of $\mathbf{A}$ is very large, the resource-constrained client wants to delegate such overload work to a public and maybe not trusted cloud server in a secure way.

In case that the system (3) has a unique solution, we can achieve the above objection by securely outsourcing the pseudoinverse of $\mathbf{A}$ modulo q. Then we can recover the unique solution $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \mod q$. However, this method requires that the matrix $\mathbf{A}$ is full column rank (*i.e.* $m \geq n$) and the product matrix $\mathbf{A}^T\mathbf{A}$ is invertible modulo q (*i.e.* $\gcd(\det(\mathbf{A}^T\mathbf{A}), q) = 1$). Also, the existing outsourcing algorithms [22, 28] for matrix inversion only detect the cloud server's misbehavior with a certain probability and thus do not achieve the optimal verifiability 1. Another direction is along the way of securely outsourcing $\mathcal{LSLE}$ [23, 29]. We can blind the inputs $\mathbf{A}$, $\mathbf{b}$ and the output $\mathbf{x}$ simultaneously by performing random sparse matrix transformations on $\mathbf{A}$ and adding a random vector $\mathbf{r}$ to $\mathbf{x}$. In other words, we can convert the Eq. (3) into

$$\mathbf{MANN}^{-1}(\mathbf{x} + \mathbf{r}) \equiv \mathbf{M}(\mathbf{b} + \mathbf{Ar}) \mod q, \tag{4}$$

where $\mathbf{M}, \mathbf{N}$ are two random sparse (or even permutation) matrices. Briefly, we denote the linear system (4) as

$$\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q, \tag{5}$$

out of which $\mathbf{A}' = \mathbf{MAN}$, $\mathbf{x}' = \mathbf{N}^{-1}(\mathbf{x}+\mathbf{r})$ and $\mathbf{b}' = \mathbf{M}(\mathbf{b}+\mathbf{Ar})$. This method can be applied for general (non-square) matrix $\mathbf{A}$ and achieve the optimal verifiability. Nonetheless, this kind of simple sparse matrix transformation can not provide strong enough privacy. E.g. it may leak the statistical information of certain entries in the matrix $\mathbf{A}$. Therefore, to design an efficient outsourcing algorithm for general matrix $\mathbf{A}$ with high security, a natural idea is to combine the above-mentioned techniques.

Different from the encryption method in [23] which encrypts $\mathbf{A}$ by only two sparse matrices, we blind the matrix $\mathbf{A}$ by dense enough matrices. Inspired by the method introduced in [22], we present a dynamic and adjustable method to encrypt the original matrix by using a series of sparse and unimodular matrix transformations. That is, the $\mathbf{A}'$ in Eq. (5) is obtained by the following method

$$\mathbf{A}' = \mathbf{U}_{f(m)} \cdots \mathbf{U}_1\mathbf{A}\mathbf{V}_1 \cdots \mathbf{V}_{g(n)}, \tag{6}$$

where $\mathbf{U}_i, \mathbf{V}_j$ are sparse and unimodular matrices for $1 \leq i \leq f(m), 1 \leq j \leq g(n)$. To ensure the efficiency of the encryption operation, the $f(m)$ (resp. $g(n)$) should be some linear function of m (resp. n) and the computation of $\mathbf{A}'$ can be efficiently implementation by virtue of the associativity of matrix multiplication. Obviously, as the increasing of the number of the sparse unimodular matrices used in the left (resp. right) transformation, the product matrix $\mathbf{U}_{f(m)} \cdots \mathbf{U}_1$ (resp. $\mathbf{V}_1 \cdots \mathbf{V}_{g(n)}$) becomes denser, and thereby the scheme performs with stronger security and lower efficiency. Therefore, the scheme provides an alternative way to balance the security and the efficiency according to different specific application scenarios.

Outsourcing scheme of $\mathbf{Ax} \equiv \mathbf{b} \mod q$

Concretely, our secure outsourcing scheme $\mathcal{SOS}_{\mathcal{LMSLE}}(q, \mathbf{A}, \mathbf{b})$ consists of the following five algorithms.

1) **Pre-processing**: Given a small constant $\lambda > 0$, this step generates a large resource pool $\mathcal{P}$ consisted of 2-by-2 random unimodular matrices as shown in equation (2).
2) **KeyGen**: On input a triple $(q, \mathbf{A}, \mathbf{b})$ with $q \in \mathbb{Z}$, $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ and $\mathbf{b} \in \mathbb{Z}_q^m$. The client C generates a random secret key $SK = (\mathbf{U}_1, \cdots, \mathbf{U}_{f(m)}, \mathbf{V}_1, \cdots, \mathbf{V}_{g(n)}, \mathbf{r})$ as follows: Firstly, this algorithm chooses some non-negative integer c which is adjustable according to different application scenarios, and defines $f(m) = c(m-3) + (c+1) + (m-2)$ and $g(n) = c(n-3) + (c+1) + (n-2)$. Secondly, for $1 \le i \le f(m)$ and $1 \le j \le g(n)$, it randomly chooses 2-by-2 unimodular matrices $\mathbf{U}^{(i)} = \begin{pmatrix} u_{11}^{(i)} & u_{12}^{(i)} \\ u_{21}^{(i)} & u_{22}^{(i)} \end{pmatrix}$ and $\mathbf{V}^{(j)} = \begin{pmatrix} v_{11}^{(j)} & v_{12}^{(j)} \\ v_{21}^{(j)} & v_{22}^{(j)} \end{pmatrix}$ from the resource pool $\mathcal{P}$. Further, if $i = 2k(m-3) + (2k+1) + (\ell-1)$ or $i = 2k(m-3) + (2k+1) - (\ell-1)$ for some $1 \le \ell \le m-1$ and $0 \le k \le c$, this algorithm generates the m-by-m unimodular matrix
$$\mathbf{U}_i = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & u_{11}^{(i)} & u_{12}^{(i)} & \cdots & 0 \\ 0 & \cdots & u_{21}^{(i)} & u_{22}^{(i)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{Z}^{m\times m}, \qquad (7)$$
by replacing the entries located in (ℓ,ℓ), $(\ell,\ell+1)$, $(\ell+1,\ell)$ and $(\ell+1,\ell+1)$ positions of the identity matrix $\mathbf{I}_{m\times m}$ with $\mathbf{U}^{(i)}$. Also, if $j = 2k(n-3) + (2k+1) + (\ell-1)$ or $j = 2k(m-3) + (2k+1) - (\ell-1)$ for some $1 \le \ell \le n-1$ and $0 \le k \le c$, this algorithm generates the n-by-n matrix
$$\mathbf{V}_j = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & v_{11}^{(j)} & v_{12}^{(j)} & \cdots & 0 \\ 0 & \cdots & v_{21}^{(j)} & v_{22}^{(j)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{Z}^{n\times n} \qquad (8)$$
by replacing the entries located in (ℓ,ℓ), $(\ell,\ell+1)$, $(\ell+1,\ell)$ and $(\ell+1,\ell+1)$ positions of the identity matrix $\mathbf{I}_{n\times n}$ with $\mathbf{V}^{(j)}$.
Finally, this step randomly generate a vector $\mathbf{r} \in \mathbb{Z}_q^n$.
3) **CEnc**: Utilizing the secret key $SK = (\mathbf{U}_1, \cdots, \mathbf{U}_{f(m)}, \mathbf{V}_1, \cdots, \mathbf{V}_{g(n)}, \mathbf{r})$, the client C encrypts the original matrix $\mathbf{A}$ and the vector $\mathbf{b}$ into
$$\mathbf{A}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1((\mathbf{A}\mathbf{V}_1) \cdots \mathbf{V}_{g(n)}))) \mod q \qquad (9)$$
and $\mathbf{b}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1(\mathbf{b} + \mathbf{Ar}))) \mod q$ respectively, and then sends $(\mathbf{A}', \mathbf{b}', q)$ to the cloud server S.
4) **SCom**: After receiving the blinded values $(\mathbf{A}', \mathbf{b}')$ and q, the cloud server S finds a solution vector $\mathbf{x}'$ such that $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$, then returns $\mathbf{x}'$ to the edge server E.
5) **EVer**: After receiving $\mathbf{x}'$, the edge server E first verifies whether the equation $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$ holds. If it does, the algorithm defines $\delta_{\mathbf{x}'} = 1$. Else, it defines $\delta_{\mathbf{x}'} = 0$. Then it sends $(\mathbf{x}', \delta_{\mathbf{x}'})$ to C.
6) **CDec**: After receiving $(\mathbf{x}', \delta_{\mathbf{x}'})$ sent from E, the client C first check the value of $\delta_{\mathbf{x}'}$. If $\delta_{\mathbf{x}'} = 1$, the algorithm outputs
$$\mathbf{x} = (\mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}'))) - \mathbf{r} \mod q\,, \qquad (10)$$
Otherwise, it outputs $\perp$.

Example 3: To make our scheme more clear, we further illustrate it with a toy example. Take $m = n = 3, f(m) = g(n) = 2, q = 7$, the original matrix
$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 6 & 0 & 5 \\ 4 & 6 & 6 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 6 \\ 0 \end{pmatrix}.$$

We omit the preprocessing step. Our scheme goes as follows.

1) The client C invokes the algorithm **KeyGen** to generate a random secret key $SK = (\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1, \mathbf{V}_2, \mathbf{r})$. Let
$$\mathbf{U}_1 = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{U}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 2 & 5 \end{pmatrix}, \mathbf{r} = \begin{pmatrix} 6 \\ 5 \\ 6 \end{pmatrix},$$
$$\mathbf{V}_1 = \begin{pmatrix} -1 & 2 & 0 \\ 2 & -3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{V}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 3 \\ 0 & 2 & -5 \end{pmatrix}$$
2) The client C invokes the algorithm **CEnc** to compute
$$\mathbf{A}' = (\mathbf{U}_2\mathbf{U}_1((\mathbf{A}\mathbf{V}_1)\mathbf{V}_2)) \mod 7 = \begin{pmatrix} 5 & 6 & 2 \\ 5 & 3 & 2 \\ 2 & 5 & 1 \end{pmatrix},$$
$$\mathbf{b}' = (\mathbf{U}_2\mathbf{U}_1(\mathbf{b} + \mathbf{Ar})) \mod 7 = \begin{pmatrix} 4 \\ 3 \\ 0 \end{pmatrix}.$$

Then C sends the three-tuple $(q, \mathbf{A}', \mathbf{b}')$ to the cloud server S.

3) The cloud sever S conducts the algorithm **SCom** to find a solution vector $\mathbf{x}'$ satisfying $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$. Namely,

$$\begin{pmatrix} 5 & 6 & 2 \\ 5 & 3 & 2 \\ 2 & 5 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 0 \end{pmatrix} \mod 7.$$

Then S returns the solution

$$\mathbf{x}' = \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \\ 4 \end{pmatrix}$$

to E.

4) The edge server E performs the algorithm **EVer** to verify whether $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$. Since it holds, E sends $(\mathbf{x}', 1)$ to C.

5) Since $\delta_{\mathbf{x}'} = 1$, the client C performs the algorithm **CDec** to decrypt

$$\mathbf{x} = (\mathbf{V}_1(\mathbf{V}_2\mathbf{x}') - \mathbf{r}) \mod 7 = \begin{pmatrix} 5 \\ 1 \\ 5 \end{pmatrix}.$$

Remarks 1 *In the preprocessing step, the set $\mathcal{P}$ can be constructed through the following method: for the 2-by-2 unimodular $\mathbf{X}$ with determinant 1, the client C randomly chooses two coprime integers $x_{11}, x_{21} \in \mathcal{X}$, and then runs the well-known extended Euclidean algorithm to compute two integers x_{12} and x_{22} subject to $x_{11}x_{22} - x_{12}x_{21} = 1$. Swapping the two columns of the matrix with determinant 1, the client C can obtain the matrix with determinant -1.*

Correctness

It is easy to see that our scheme is correct for our toy example. Now, we prove the correctness of the proposed scheme for arbitrary instance.

Theorem 1 *For any valid inputs $q \in \mathbb{Z}, \mathbf{b} \in \mathbb{Z}_q^m$, and $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, the proposed secure outsourcing scheme $\mathcal{SOS}_{\mathcal{LMSLE}}(q, \mathbf{A}, \mathbf{b})$ is correct according to the definition 1.*

Proof For the inputs $q \in \mathbb{Z}, \mathbf{b} \in \mathbb{Z}_q^m$, assume that the *SK* produced by the algorithm **KeyGen** is $(\mathbf{U}_1, \cdots, \mathbf{U}_{f(m)}, \mathbf{V}_1, \cdots, \mathbf{V}_{g(n)}, \mathbf{r})$. Then, invoking the encryption algorithm **CEnc**,

$$\mathbf{A}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1((\mathbf{A}\mathbf{V}_1) \cdots \mathbf{V}_{g(n)}))) \mod q$$

and $\mathbf{b}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1(\mathbf{b} + \mathbf{A}\mathbf{r}))) \mod q$. If the result $\mathbf{x}'$ computed by the cloud server satisfies $\mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q$, the algorithm **EVer** sends $(\mathbf{x}', 1)$ to the client C. Therefore, the algorithm **CDec** outputs

$$\mathbf{x} = \mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}')) - \mathbf{r} \mod q\,,$$

which satisfies

$$\begin{aligned}
&\mathbf{A}\mathbf{x} = \mathbf{A}\left(\mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}')) - \mathbf{r} \mod q\right) \\
&= \mathbf{A}\mathbf{V}_1\mathbf{V}_2 \cdots \mathbf{V}_{g(n)} \cdot \mathbf{x}' - \mathbf{A}\mathbf{r} \mod q \\
&= (\mathbf{U}_{f(m)} \cdots \mathbf{U}_1)^{-1}(\mathbf{U}_{f(m)} \cdots \mathbf{U}_1)\mathbf{A}\mathbf{V}_1\mathbf{V}_2 \cdots \mathbf{V}_{g(n)} \cdot \mathbf{x}' \\
&- \mathbf{A}\mathbf{r} \mod q \\
&= (\mathbf{U}_{f(m)} \cdots \mathbf{U}_1)^{-1}\mathbf{A}'\mathbf{x}' - \mathbf{A}\mathbf{r} \mod q \\
&= (\mathbf{U}_{f(m)} \cdots \mathbf{U}_1)^{-1}\mathbf{b}' - \mathbf{A}\mathbf{r} \mod q \\
&= (\mathbf{U}_{f(m)} \cdots \mathbf{U}_1)^{-1}(\mathbf{U}_{f(m)} \cdots \mathbf{U}_1(\mathbf{b} + \mathbf{A}\mathbf{r})) - \mathbf{A}\mathbf{r} \mod q \\
&= \mathbf{b} + \mathbf{A}\mathbf{r} - \mathbf{A}\mathbf{r} \mod q \\
&= \mathbf{b} \mod q.
\end{aligned}$$

Namely, $\mathbf{x} = (\mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}'))) - \mathbf{r} \mod q$ is a solution of $\mathbf{A}\mathbf{x} \equiv \mathbf{b} \mod q$. That is, the algorithm **CDec**$(\mathbf{x}', SK)$ outputs correct result. □

Input/output privacy

Now, following the analysis of the work [23], we prove that the input $(\mathbf{A}, \mathbf{b})$ and the actual output $\mathbf{x}$ is private against the *FM* cloud server and the *HBC* edge server in our scheme.

Theorem 2 *For any valid inputs $q \in \mathbb{Z}, \mathbf{b} \in \mathbb{Z}_q^m$, $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $(\mathbf{A}, \mathbf{b})$ (resp. $\mathbf{x}$) in our proposed secure outsourcing algorithm $\mathcal{SOS}_{\mathcal{LMSLE}}(q, \mathbf{A}, \mathbf{b})$ satisfies the input (resp. output) privacy according to the Definition 2.*

Proof First, we prove the privacy for the input vector $\mathbf{b}$ and the output vector $\mathbf{x}$. Since $\mathbf{b} = \mathbf{U}_1^{-1} \cdots \mathbf{U}_{f(m)}^{-1}\mathbf{b}' - \mathbf{A}\mathbf{r} \mod q$, $\mathbf{x} = (\mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}'))) - \mathbf{r} \mod q$, $\mathbf{r} \in \mathbb{Z}_q^n$ is random and $\mathbf{A}$ is full rank, both $\mathbf{b}$ and $\mathbf{x}$ are blinded by $\mathbf{r}$ in sense of indistinguishability. That is, the cloud/edge server has to guess the value of $\mathbf{b}$ (resp. $\mathbf{x}$), and then the probability is

$$\frac{1}{\#\mathbb{Z}_q^m} = \frac{1}{q^m} \quad \left(resp.\ \frac{1}{\#\mathbb{Z}_q^n} = \frac{1}{q^n}\right),$$

which clearly is negligible function of m (resp.n).

$\mathbf{A}$-privacy: If the cloud/edge server wants to steal the information of the original coefficient matrix $\mathbf{A}$, since it knows the ciphertext matrix

$$\mathbf{A}' = \mathbf{U}_{f(m)} \cdots \mathbf{U}_1\mathbf{A}\mathbf{V}_1 \cdots \mathbf{V}_{g(n)} \mod q,$$

it can recover $\mathbf{A}$ by computing

$$\mathbf{A} = \mathbf{U}_1^{-1} \cdots \mathbf{U}_{f(m)}^{-1}\mathbf{A}'\mathbf{V}_{g(n)}^{-1} \cdots \mathbf{V}_1^{-1} \mod q.$$

Let $\mathbf{U} = \mathbf{U}_1^{-1} \cdots \mathbf{U}_{f(m)}^{-1}$ and $\mathbf{V} = \mathbf{V}_{g(n)}^{-1} \cdots \mathbf{V}_1^{-1}$. Then

$$a_{ij} = \sum_{s=1}^{m} \sum_{t=1}^{n} u_{is} a'_{st} v_{tj} \mod q.$$

Since u_{is}, v_{tj} are random, a_{ij} is also indistinguishable with a random number in $\mathbb{Z}_q$. The cloud/edge server can guess a_{ij} by brute-force attack, in which case the probability of recovering $\mathbf{A} = (a_{ij})_{1 \le i \le m, 1 \le j \le n}$ is

$$\frac{1}{\#\mathbb{Z}_q^{m+n}} = \frac{1}{q^{m+n}}.$$

It is clearly a negligible function of m (resp. n).

Another feasible way of recovering $\mathbf{A}$ is to obtain the unimodular matrices $\mathbf{U}_i$ and $\mathbf{V}_j$ for $1 \le i \le f(m)$ and $1 \le j \le g(n)$. Since each $\mathbf{U}_i$ (resp. $\mathbf{V}_j$) is constructed by a 2-by-2 unimdoular matrix randomly chosen from a set $\mathcal{P}$ with size $\#\mathcal{P} \ge \frac{48}{\pi^2}(2^\lambda - 1)^2$, the probability of the cloud/edge server can guess the correct $\mathbf{U}_i$ (resp.$\mathbf{V}_j$) is

$$\frac{1}{\#\mathcal{P}} \le \frac{1}{\frac{48}{\pi^2}(2^\lambda - 1)^2},$$

Overall, the probability of the malicious cloud server can guess the correct $\mathbf{A}$ is

$$\left(\frac{1}{\#\mathcal{P}}\right)^{f(m)+g(n)} \le \left(\frac{1}{\frac{48}{\pi^2}(2^\lambda - 1)^2}\right)^{f(m)+g(n)},$$

which also is negligible function of m (resp. n). □

Verifiability

In this section, we will prove that the edger server and thus the client can detect the cloud server's misbehaviours with probability 1. Strictly speaking, we have the following result.

Theorem 3 *For any valid inputs $q \in \mathbb{Z}, \mathbf{b} \in \mathbb{Z}_q^m$, $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, the proposed secure outsourcing algorithm $\mathcal{SOS}_{\mathcal{LMSLE}}(\mathbf{A}, q, \mathbf{b})$ is 1-verifiability according to the Definition 3.*

Proof Based on the Definition 3, we need to prove the following two conditional probability identities:

$$\Pr[\,\delta_{\mathbf{x}'} = 1 \leftarrow \mathbf{EVer}(\mathbf{x}', \Phi') \mid \mathbf{A}'\mathbf{x}' \equiv \mathbf{b}' \mod q] = 1, \quad (11)$$
$$\Pr[\,\delta_{\mathbf{x}'} = 1 \leftarrow \mathbf{EVer}(\mathbf{x}', \Phi') \mid \mathbf{A}'\mathbf{x}' \not\equiv \mathbf{b}' \mod q] = 0. \quad (12)$$

The identity (11) is straightforwardly from the correctness of our scheme, and the identity (12) is also obviously from the definition of algorithm $\mathbf{EVer}(\mathbf{x}', \Phi')$. □

Efficiency

Let $t_{\mathbf{KeyGen}}, t_{\mathbf{CEnc}}, t_{\mathbf{SCom}}, t_{\mathbf{EVer}}$ and $t_{\mathbf{CDec}}$ denote the time overhead of corresponding algorithm in our scheme.

Overhead of **KeyGen**. In the key generation step, the secret key $\mathcal{SK} = (\mathbf{U}_1, \cdots, \mathbf{U}_{f(m)}, \mathbf{V}_1, \cdots, \mathbf{V}_{g(n)}, \mathbf{r})$ is generated by randomly chosen operation. Since the size of key space $\#\mathcal{P} \le 2^{8\lambda}$ and the size of $\mathbb{Z}_q^n$ is q^n, the time overhead of this step is $t_{\mathbf{KeyGen}} = O((f(m) + g(n))\lambda + n \log q)$.

Overhead of **CEnc**. In the encryption step, the client can compute $\mathbf{A}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1((\mathbf{A}\mathbf{V}_1) \cdots \mathbf{V}_{g(n)}))) \mod q$, and $\mathbf{b}' = (\mathbf{U}_{f(m)} \cdots (\mathbf{U}_1(\mathbf{b} + \mathbf{A}\mathbf{r}))) \mod q$ by associativity. Therein, computing $\mathbf{A}'$ requires $4mg(n) + 4nf(m)$ multiplications over ring $\mathbb{Z}_q$, and computing $\mathbf{b}'$ requires $mn + 4f(m)$ multiplications over ring $\mathbb{Z}_q$. Since the multiplication operation is performed on two integers with bit length no more than $\log q$, the time overhead of this step is $t_{\mathbf{CEnc}} = O((mg(n) + nf(m) + mn)(\log q)^2)$.

Overhead of **SCom**. In the cloud server computation step, C needs to solve the large-scale modular system of linear equations $\mathbf{A}'\mathbf{x}' \mod q$ which has a time complexity of $t_{\mathbf{SCom}} = O(mn^2(\log q)^2)$.

Overhead of **EVer**. In the edge server verification step, E also needs to compute $\mathbf{A}'\mathbf{x}' \mod q$, which requires mn multiplications over ring $\mathbb{Z}_q$. Also, the multiplication operation is performed on two integers with bit length no more than $\log q$. Therefore, $t_{\mathbf{EVer}} = t_{\mathbf{SCom}} = O(mn(\log q)^2)$.

Overhead of **CDec**. In the client decryption step, the client needs to compute $\mathbf{x} = (\mathbf{V}_1(\mathbf{V}_2 \cdots (\mathbf{V}_{g(n)} \cdot \mathbf{x}'))) - \mathbf{r} \mod q$,which requires $4g(n)$ multiplications over ring $\mathbb{Z}_q$. Also, the multiplication operation is performed on two integers with bit length no more than $\log q$. Therefore, $t_{\mathbf{CDec}} = O(g(n)(\log q)^2)$.

Since $f(m)$ (resp. $g(n)$) is a linear function of m (resp.n) and λ is a small constant, the overall time overhead on the client side is

$$\begin{aligned} t_{\text{client}} &= t_{\mathbf{KeyGen}} + t_{\mathbf{CEnc}} + t_{\mathbf{CDec}} \\ &= O((mg(n) + nf(m) + mn)(\log q)^2) \\ &= O(mn(\log q)^2). \end{aligned}$$

The time overhead of the cloud server S is

$$t_{\text{cloud}} = t_{\mathbf{SCom}} = O(mn^2(\log q)^2),$$

and the time overhead of the edge server E is

$$t_{\text{edge}} = t_{\mathbf{EVer}} = O(mn(\log q)^2).$$

Since, if the client performs the work by itself, the time overhead is $t_o = O(mn^2(\log q)^2)$. Therefore efficiency in Definition 4 is

$$\alpha = \frac{t_{\text{original}}}{t_{\text{client}}} = O\left(\frac{mn^2(\log q)^2}{mn(\log q)^2}\right) = O(n).$$

That is, we have proved the following result

Theorem 4 *For any valid inputs $q \in \mathbb{Z}, \mathbf{b} \in \mathbb{Z}_q^m$, $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, the proposed secure outsourcing algorithm $\mathcal{SOS}_{\mathcal{LMSLE}}(\mathbf{A}, q, \mathbf{b})$ is $O(n)$-Efficiency according to the Definition 4.*

Performance evaluation

To evaluate the practical performance of our scheme, in this section, we provide a comprehensive experimental analysis.

Evaluation methodology

Our analysis is presented by experimentally simulating the outsourcing process. All the experiments are implemented on Mathmatica 11.0 software platform. For the algorithms performed on the client side, we carry out them on a computer with Intel i3-2330M CPU processor running at 2.20 GHz, 2 GB memory. We simulate the edge server on a computer with Intel(R) Core(TM) i5-7200U CPU processor running at 2.71 GHz, 8 GB memory, and simulate the cloud server on a computer with Intel(R)Xeon(R) W-2123 CPU @ 3.60GHz 3.60 GHz, 32.0 GB memory.

To comprehensively evaluate the performance of our scheme, we make comparisons from the following three different perspectives:

- Compare the client's time cost of outsourcing different computation instances by performing the proposed scheme with that of solving the corresponding computation instances on its own.
- Compare the total time cost of outsourcing different computation instances by performing the proposed scheme with that of achieving the corresponding computation instances on client's own.
- Simulate the variance of the client's time cost of outsourcing different computation instances with different $f(m)$ and $g(n)$ in our scheme.

Evaluation results

Let t_{original}, $t_{\mathbf{CEnc}}$, $t_{\mathbf{CDec}}$, $t_{\mathbf{SCom}}$, $t_{\mathbf{Ever}}$, t_{client}, t_{cloud}, t_{edge} denote the same meanings as that in "Efficiency" section. $t_{\text{totall}(\Sigma)} = t_{\text{client}} + t_{\text{cloud}} + t_{\text{edge}}$ represents the total cost of our scheme for solving $\mathcal{LMSLE}$ instances. We take $q = 251$ and $q = 12289$ which are common in learning with errors (LWE)-based cryptosystems [19, 30]. The parameter λ is set to 2 in our experiments. The experimental results are shown in Tables 1, 2, 3, 4, 5, 6, 7 and 8. Noting that the time overhead of the algorithm **KeyGen** is negligible compared with other algorithms, we omit it in the tables.

From the tables, we can easily observed the following three results: (1) Our proposed scheme enables the client to achieve decent computational savings compared with conducting the task on client's own, and the speedup ratio

Table 1 $\mathbf{Ax} \equiv \mathbf{b} \mod 251$ with parameters $f(m) = m - 1, g(n) = n - 1$ (unit:s)

	Problem Size	$t_{\mathbf{CEnc}}$	$t_{\mathbf{SCom}}(t_{\text{cloud}})$	$t_{\mathbf{EVer}}(t_{\text{edge}})$	$t_{\mathbf{CDec}}$
1	450×500	0.07587596	0.7940	0.00177463	0.00950239
2	950×1000	0.17872325	6.21088	0.00844535	0.0125903
3	1450×1500	0.3961571	20.783	0.0184996	0.0277531
4	1950×2000	0.6928951	49.303	0.0329889	0.0290819
5	2450×2500	1.2596379	95.0151	0.0489073	0.0454629
6	2950×3000	1.7189	161.221	0.0731548	0.0451746

Table 2 $\mathbf{Ax} \equiv \mathbf{b} \mod 251$ with parameters $f(m) = m - 1, g(n) = n - 1$ (Continue with Table 1, unit:s)

	Problem Size	t_{client}	$t_{\text{totall}(\Sigma)}$	t_{original}	$\frac{t_{\text{original}}}{t_{\text{client}}}$	$\frac{t_{\text{original}}}{t_{\text{totall}(\Sigma)}}$
1	450×500	0.08537835	0.88115298	1.40008	16.40	1.5889
2	950×1000	0.19131355	6.4106389	10.5693	55.25	1.6487
3	1450×1500	0.4239102	21.2254098	36.6499	86.46	1.7267
4	1950×2000	0.721977	50.0579659	86.9538	120.44	1.7371
5	2450×2500	1.3051008	96.3691081	169.814	130.12	1.7621
6	2950×3000	1.7640746	163.058229	293.245	166.23	1.7984

Table 3 $\mathbf{Ax} \equiv \mathbf{b} \mod 251$ with parameters $f(m) = 2m - 3, g(n) = 2n - 3$ (unit:s)

	Problem Size	$t_{\mathbf{CEnc}}$	$t_{\mathbf{SCom}}(t_{\text{cloud}})$	$t_{\mathbf{EVer}}(t_{\text{edge}})$	$t_{\mathbf{CDec}}$
1	450×500	0.10514155	0.7908	0.0017516	0.0192382
2	950×1000	0.34959892	6.208	0.00861224	0.0395156
3	1450×1500	0.7866935	20.783	0.0182387	0.0443406
4	1950×2000	1.3715709	49.303	0.032109	0.0596953
5	2450×2500	2.3783475	95.0151	0.0488523	0.0828955
6	2950×3000	3.4255175	160.891	0.0721897	0.0802106

Table 4 $\mathbf{Ax} \equiv \mathbf{b} \mod 251$ with parameters $f(m) = 2m - 3, g(n) = 2n - 3$ (Continue with Table 3, unit:s)

	Problem Size	t_{client}	$t_{\text{totall}(\Sigma)}$	t_{original}	$\frac{t_{\text{original}}}{t_{\text{client}}}$	$\frac{t_{\text{original}}}{t_{\text{totall}(\Sigma)}}$
1	450×500	0.12437975	0.91693135	1.34423	10.81	1.466
2	950×1000	0.38911452	6.60572676	10.9535	28.15	1.6582
3	1450×1500	0.8310341	21.6322728	36.8087	44.25	1.70156
4	1950×2000	1.4312662	50.7663752	86.9632	60.76	1.713
5	2450×2500	2.461243	97.5251953	170.006	69.07	1.7432
6	2950×3000	3.5057281	164.468918	293.878	85.82	1.78683

Table 5 $\mathbf{Ax} \equiv \mathbf{b} \mod 12289$ with parameters $f(m) = m - 1, g(n) = n - 1$ (unit:s)

	Problem Size	$t_{\mathbf{CEnc}}$	$t_{\mathbf{SCom}}(t_{\text{cloud}})$	$t_{\mathbf{EVer}}(t_{\text{edge}})$	$t_{\mathbf{CDec}}$
1	450×500	0.06575108	0.8125	0.00202232	0.00749616
2	950×1000	0.17575052	6.05226	0.00890071	0.0131726
3	1450×1500	0.3993411	20.2893	0.0201133	0.0278017
4	1950×2000	0.6892857	48.2628	0.0346034	0.0350569
5	2450×2500	1.2048372	95.8165	0.0498146	0.0475414
6	2950×3000	1.7189946	161.006	0.0748025	0.0497302

Table 6 $\mathbf{Ax} \equiv \mathbf{b} \mod 12289$ with parameters $f(m) = m - 1, g(n) = n - 1$ (Continue with Table 5,unit:s)

	Problem Size	t_{client}	$t_{\text{totall}(\Sigma)}$	t_{original}	$\frac{t_{\text{original}}}{t_{\text{client}}}$	$\frac{t_{\text{original}}}{t_{\text{totall}(\Sigma)}}$
1	450×500	0.07324724	0.88776956	1.40402	19.17	1.5815
2	950×1000	0.18892312	6.25008383	11.121	58.87	1.7793
3	1450×1500	0.4271428	20.7365561	37.3979	87.56	1.803
4	1950×2000	0.7243426	49.021746	89.3161	123.31	1.821
5	2450×2500	1.2523786	97.1186932	172.927	138.08	1.7806
6	2950×3000	1.7687248	162.849527	299.213	169.17	1.83736

Table 7 $\mathbf{Ax} \equiv \mathbf{b} \mod 12289$ with parameters $f(m) = 2m - 3, g(n) = 2n - 3$(unit:s)

	Problem Size	$t_{\mathbf{CEnc}}$	$t_{\mathbf{SCom}}(t_{\text{cloud}})$	$t_{\mathbf{EVer}}(t_{\text{edge}})$	$t_{\mathbf{CDec}}$
1	450×500	0.09749518	0.8022	0.00210615	0.0131503
2	950×1000	0.33596836	6.1038	0.00890071	0.0252105
3	1450×1500	0.7694877	20.2703	0.0195651	0.0406829
4	1950×2000	1.3436429	48.9895	0.034724	0.0653062
5	2450×2500	2.3465147	95.0048	0.049123	0.0964595
6	2950×3000	3.3026583	161.087	0.074192	0.0948513

Table 8 $\mathbf{Ax} \equiv \mathbf{b} \mod 12289$ with parameters $f(m) = 2m - 3, g(n) = 2n - 3$ (Continue with Table 7, unit:s)

	Problem Size	t_{client}	$t_{\text{totall}(\Sigma)}$	t_{original}	$\frac{t_{\text{original}}}{t_{\text{client}}}$	$\frac{t_{\text{original}}}{t_{\text{totall}(\Sigma)}}$
1	450×500	0.11064548	0.91495163	1.40206	12.67	1.5324
2	950×1000	0.36117886	6.47387957	11.0904	30.71	1.7131
3	1450×1500	0.8101706	21.1000357	37.5176	46.31	1.7781
4	1950×2000	1.4089491	50.4331731	88.3165	62.68	1.7511
5	2450×2500	2.4429742	97.4968972	172.665	70.68	1.771
6	2950×3000	3.3975096	164.558702	298.048	87.73	1.8112

$(\frac{t_{\text{original}}}{t_{\text{client}}})$ increases monotonously with the size of the input instances. For example, when $q = 251, f(m) = m - 1, g(n) = n-1$, the size of coefficient matrix is 450×500, the client obtains 16.40 times speedup in efficiency, and, when the size of coefficient matrix increases to 2950×3000, the speedup ratio achieves 166.23. (2) The total time overhead of our scheme is cheaper than that of conducting corresponding computation instances on client's own, and the speedup ratio $(\frac{t_{\text{original}}}{t_{\text{total}(\Sigma)}})$ also increases with the increment of the problem size. For instance, when $q = 12289$, $f(m) = m-1, g(n) = n-1$, the size of coefficient matrix is 450×500, our scheme obtains 1.58 times speedup in efficiency, and, when the size of coefficient matrix increases to 2950×3000, the speedup ratio achieves 1.83. (3) The efficiency/security of our scheme is adjustable and the efficiency decreases with the increment of $f(m)$ and $g(n)$. E.g. when $q = 251$ with $f(m) = m - 1, g(n) = n - 1$, and the problem size is 2950×3000, the client's time overhead can be reduced by $99.40\% = (1 - 1/166) * 100\%$ by outsourcing solving $\mathbf{Ax} \equiv \mathbf{b} \mod 251$, while, when the same problem size with $f(m) = 2m - 3$, $g(n) = 2n - 3$, the client's time cost is only reduced by $98.82\% = (1 - 1/85) * 100\%$.

As we have observed from the experimental results, the proposed scheme is effective and achieves great benefits in efficiency compared with the algorithm without outsourcing.

Related work

Since a wide range of applications of linear algebraic operations in various fields, outsourcing linear algebraic operations, such as matrix multiplication computation (MMC) [31], matrix determinant computation (MDC) [32], matrix factorization [33–35], and matrix's characteristic polynomial and eigenvalues computation [36] has become a hot topic [8]. Out of which, the closely related operations with our work are matrix inversion computation (MIC) and large-scale system of linear equations $(\mathcal{LSLE})$solving.

Matrix inversion computation outsourcing. In 2012, Mohassel first investigated the secure outsourcing of MIC and initialized a secure outsourcing algorithm [20]. The algorithm encrypts the original matrix through a random matrix transformation technique (i.e. multiplying the original matrix by a random secret matrix) and invokes the outsourcing algorithm of MMC as a subroutine. Since the outsourcing algorithm of MMC needs to perform expensive homomorphic encryption (HE) operations. Therefore, their algorithm suffers from low efficiency and then is not practical. To avoid the time-consuming HE operations, Lei et al. [21] further put forward a new protocol by directly exploiting a simple random permutation matrix transformation technique.

However, this technique is inherently with a severe security problem of exposing the statistical information of some certain entries in the original matrix, e.g. the number of zeros. Recently, Zhang et al. [22] presented a novel matrix encryption method based on consecutive sparse and unimodular matrix transformation, and then exploited this method to design a new MIC outsourcing algorithm over finite fields. Their algorithm favorably balances the security and efficiency, i.e. it conceals the entries' statistical information in the original matrix without greatly reducing the efficiency. However, the verification algorithm of all the aforementioned schemes is based on a randomized Monte Carlo verification, which is not deterministic with the optimal probability 1. Also, the outsourcing methods of MIC are only suit for outsourcing $\mathcal{LMSLE}$ with a square and invertible coefficient matrix $\mathbf{A}$.

$\mathcal{LSLE}$ computation outsourcing. Atallah et al. [37] initialized the study on secure outsourcing of linear algebraic operations, and presented the first efficient outsourcing scheme for solving $\mathcal{LSLE}$ by using simple random permutation matrix transformation. Nevertheless, just as mentioned in Lei et al.'s work [21], this technique leaks the statistic information of entries in the coefficient matrix. Also, their algorithm doesn't concern the result verification. Wang et al. [38] further investigated this problem and proposed a privacy-preserving, cheating-resilient and effective outsourcing protocol based on iterative method. However, their protocol needs to invoke the expensive homomorphic encryption scheme [39] and requires multi-round interactions between the client and the cloud server, which incurs large communication and computation overhead. Noticing this deficiency of Wang et al.'s protocol, Chen et al. [40] proposed an improved scheme based on a new matrix encryption method which multiplies the coefficient matrix by diagonal and random permutation matrices. Afterward, Chen et al. [23] put forward a new secure outsourcing algorithm for solving $\mathcal{LSLE}$ based on a sparse matrix transformation technique. These sparse matrix encryption methods make their algorithms achieve high efficiency with optimal communication overhead and optimal verifiability probability 1. Almost at the same time, Salinas et al. [41] proposed an efficient outsourcing algorithm based on the conjugate gradient method which achieves low computational complexity and low memory I/O complexity. Nonetheless, the algorithm also needs multi-round interactions and is not security as they claimed. Subsequently, based on Salinas et al.'s matrix encryption method, Yu et al. [42] proposed a secure non-iterative outsourcing algorithm for solving $\mathcal{LSLE}$, but the algorithm calls the outsourcing algorithm of MMC as a subroutine which leads to be inefficient in practice. Recently, Ding et al. [43] successfully attacked Salinas et al.'s scheme and presents an efficient algorithm to recover the protected matrix. Therefore, among these proposals, Chen et al.'s [23] scheme is the most efficient, but, just as the authors' mentioned in their paper, the sparse matrix transformation can not provide strong enough security.

Conclusion and future direction

In this paper, we introduce an efficiency/security-adjustable scheme for publicly verifiable delegation of computation which enable a resource-constrained client to securely outsource the solving of $\mathcal{LMSLE}$. We built our scheme upon a novel dynamic affine transformation by successively utilizing sparse and unimodular matrices which may has potential applications in outsourcing other linear algebraic operations. However, there still exist some open problems deserved for further research. First, in the case that the $\mathcal{LMSLE}$ problem has multiple solutions, the presented outsourcing scheme can only ensure the client to securely find one solution. That is, if the cloud server returns multiple solutions, the client or the edge server in our scheme can verify the correctness of each solution, but it can not confirm whether the cloud returns the set of all solutions. Therefore, how to design an efficient outsourcing scheme that enables the client to securely and correctly find the set of all solutions is an interesting problem. Moreover, for some strong threat model including outside adversary [44], our scheme may suffer from unauthorized attack. It is meaningful to improve our scheme to resist this attack. Adding a simple identity authentication protocol may be a good choice. Also, our scheme does not consider the privacy preservation of the modulus q. In some practical application scenarios, the modulus may contain sensitive information, such as the bit length of the solution vector, and thus keeping the privacy of the modulus is also very significative. Consequently, how to design a modulus-invisible outsourcing scheme for solving $\mathcal{LMSLE}$ is also deserved for further research.

Acknowledgement
Not Applicable.

Authors' contributions
The authors equally contributed to this research and the paper initiated by the corresponding author. All authors read and approved the final manuscript.

Authors' information
Panpan Meng received the B.E. degree in computer information management from Shandong University in 2016. She is currently pursuing the M.S. degree in the College of Computer Science and Technology, Qingdao University. Her research interests include cloud computing security and cryptography.
Chengliang Tian received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 2006 and 2009, respectively, and the Ph.D. degree in information security from Shandong University, Ji'nan, China, in 2013. He held a post-doctoral position with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing. He is currently with the College of Computer Science and Technology, Qingdao University, as an Assistant Professor. His research interests include lattice-based cryptography and cloud computing security.

Funding
This research is supported in part by National Natural Science Foundation of China (61702294), Natural Science Foundation of Shandong Province (ZR2016FQ02), National Development Foundation of Cryptography (MMJJ20170126), the Open Research Project (2016-MS-23) of State Key Laboratory of Information Security in Institute of Information Engineering, Chinese Academy of Sciences, Applied Basic Research Project of Qingdao City (17-1-1-10-jch).

Availability of data and materials
The full experimental data and the description of the experiment setup are provided in the manuscript in section `Performance Evaluation`.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China. [2]Business School, Qingdao University, Qingdao 266071, China. [3]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China.

Received: 15 July 2019 Accepted: 20 November 2019
Published online: 27 December 2019

References

1. Singh KJ, Kapoor DS (2017) Create your own internet of things: A survey of iot platforms. IEEE Consum Electron Mag 6(2):57–68
2. Xu LD, He W, Li S (2014) Internet of things in industries: A survey. IEEE Trans Ind Informa 10(4):2233–2243
3. Xia H, Zhang S, Li Y, Pan Z, Peng X, Cheng X (2019) An attack-resistant trust inference model for securing routing in vehicular ad hoc networks. IEEE Trans Veh Technol 68(7):7108–7120
4. Rivera J, Goasduff L (2014) Gartner says a thirty-fold increase in internet-connected physical devices by 2020 will significantly alter how the supply chain operates. Gartner. https://www.gartner.com/en/newsroom/press-releases/2014-03-24-gartner-says-a-thirty-fold-increase-in-internet-connected-physical-devices-by-2020-will-significantly-alter-how-the-supply-chain-operates
5. Brunette G, Mogull R, et al. (2009) Security guidance for critical areas of focus in cloud computing v2. 1. Cloud Security Alliance. http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf
6. Xia H, Cq Hu, Xiao F, Cheng Xg, Pan Zk (2019) An efficient social-like semantic-aware service discovery mechanism for large-scale internet of things. Comput Netw 152:210–220
7. Xia H, Zhang Ss, Li Bx, Li L, Cheng Xg (2018) Towards a novel trust-based multicast routing for vanets. Secur Commun Netw 2018:1–1
8. Shan Z, Ren K, Blanton M, Wang C (2018) Practical secure computation outsourcing: A survey. Acm Comput Surv 51(2):1–40
9. Zhou Q, Tian C, Zhang H, Yu J, Li F (2019) How to securely outsource the extended euclidean algorithm for large-scale polynomials over finite fields. Inf Sci. https://doi.org/10.1016/j.ins.2019.10.007
10. von zur Gathen J, Gerhard J (2013) Modern Computer Algebra. 3rd edition. Cambridge University Press, New York
11. Moon TK (2005) Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience, New York
12. Ryan W, Lin S (2009) Channel Codes: Classical and Modern. Cambridge University Press, New York
13. Ajtai M (1996) Generating hard instances of lattice problems (extended abstract). In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, STOC '96. ACM, New York. pp 99–108
14. Gentry C, Peikert C, Vaikuntanathan V (2008) Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th annual ACM symposium on Theory of computing, STOC '08. ACM, New York. pp 197–206
15. Peikert C (2014) Lattice cryptography for the internet. In: Mosca M (ed). Post-Quantum Cryptography. Springer International Publishing, Cham. pp 197–219
16. Gentry C, Halevi S, Vaikuntanathan V (2010) A simple bgn-type cryptosystem from lwe. In: Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT'10. Springer-Verlag, Berlin, Heidelberg. pp 506–522
17. Regev O (2005) On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, STOC '05. ACM, New York. pp 84–93
18. Bos JW, Costello C, Naehrig M, Stebila D (2015) Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In: 2015 IEEE Symposium on Security and Privacy. IEEE, San Jose. pp 553–570
19. Alkim E, Ducas L, Pöppelmann T, Schwabe P (2016) Post-quantum key exchange-a new hope. In: 25th USENIX Security Symposium (USENIX Security 16). USENIX Association, Austin. pp 327–343
20. Mohassel P (2012) Efficient and secure delegation of linear algebra. Iacr Cryptol Eprint Archive. https://eprint.iacr.org/2011/605
21. Lei X, Liao X, Huang T, Li H, Hu C (2013) Outsourcing large matrix inversion computation to a public cloud. IEEE Trans Cloud Comput 1(1):1–1
22. Zhang S, Tian C, Zhang H, Yu J, Li F (2019) Practical and secure outsourcing algorithms of matrix operations based on a novel matrix encryption method. IEEE Access 7:53823–53838
23. Chen X, Huang X, Li J, Ma J, Lou W, Wong DS (2015) New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Trans Inf Forensic Secur 10(1):69–78
24. Newman M (1972) Integral matrices, volume 45. Academic Press, New York
25. Horn RA, Johnson CR (2012) Matrix Analysis. 2nd edition. Cambridge University Press, New York
26. Schrijver A (1986) Theory of Linear and Integer Programming. Wiley, New York
27. Apostol TM (1998) Introduction to Analytic Number Theory. Undergraduate Texts in Mathematics. Springer, New York
28. Lei X, Liao X, Huang T, Heriniaina F (2014) Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. Inf Sci 280:205–217
29. Qian C, Wang J (2015) Secure and efficient protocol for outsourcing large-scale systems of linear equations to the cloud. In: International Conference on Cloud Computing and Security. Springer, Basel. pp 25–37
30. Lu X, Liu Y, Zhang Z, Jia D, Xue H, He J, Li B, Wang K, Liu Z, Yang H (2018) Lac: Practical ring-lwe based public-key encryption with byte-level modulus. Cryptology ePrint Archive, Report 2018/1009. https://eprint.iacr.org/2018/1009. Accessed 16 Oct 2018
31. Atallah MJ, Frikken KB (2010) Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10. ACM, New York. pp 48–59
32. Lei X, Liao X, Huang T, Li H (2015) Cloud computing service: The case of large matrix determinant computation. IEEE Trans Serv Comput 8(5):688–700
33. Zhou L, Zhu Y, Choo K-KR (2018) Efficiently and securely harnessing cloud to solve linear regression and other matrix operations. Futur Gener Comput Syst 81:404–413
34. Luo C, Zhang K, Salinas S, Li P (2017) Efficient privacy-preserving outsourcing of large-scale qr factorization. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, Sydney. pp 917–924
35. Luo C, Zhang K, Salinas S, Li P (2017) Secfact: Secure large-scale qr and lu factorizations. IEEE Trans Big Data. https://ieeexplore.ieee.org/document/8194901
36. Hu X, Tang C (2015) Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix. J Cloud Comput 4(1):7
37. Atallah MJ, Pantazopoulos KN, Rice JR, Spafford EE (2002) Secure outsourcing of scientific computations. In: Advances in Computers, volume 54. Elsevier, Cambridge. pp 215–272
38. Wang C, Ren K, Wang J, Wang Q (2013) Harnessing the cloud for securely outsourcing large-scale systems of linear equations. IEEE Trans Parallel Distrib Syst 24(6):1172–1181
39. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed). Advances in Cryptology — EUROCRYPT '99. Springer Berlin Heidelberg, Berlin, Heidelberg. pp 223–238
40. Chen F, Xiang T, Yang Y (2014) Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud. J Parallel Distrib Comput 74(3):2141–2151
41. Salinas S, Luo C, Chen X, Li P (2015) Efficient secure outsourcing of large-scale linear systems of equations. In: Computer Communications (INFOCOM) 2015 IEEE Conference on. IEEE, Atlanta. pp 1035–1043

42. Yu Y, Luo Y, Wang D, Fu S, Xu M (2016) Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations. In: 2016 IEEE International Conference on Communications (ICC). pp 1–6
43. Ding Q, Weng G, Zhao G, Hu C (2018) Efficient and secure outsourcing of large-scale linear system of equations. IEEE Trans Cloud Comput:1–1. https://ieeexplore.ieee.org/document/8531754
44. Wu J, Mu N, Lei X, Le J, Zhang D, Liao X (2019) Secedmo: Enabling efficient data mining with strong privacy protection in cloud computing. IEEE Trans Cloud Comput:1–1. https://ieeexplore.ieee.org/document/8781873

Publisher's Note