Weave ElGamal Encryption for Secure Outsourcing Algebraic Computations over $\mathbb{Z}_p$*

Yi-Ruei Chen, Shiuan-Tzuo Shen, and Wen-Guey Tzeng

Department of Computer Science, National Chiao Tung University, Taiwan
yrchen.cs98g@nctu.edu.tw, vink@cs.nctu.edu.tw, wgtzeng@cs.nctu.edu.tw

Abstract. This paper addresses the secure outsourcing problem for large-scale matrix computation to a public cloud. We propose a novel public-key *weave ElGamal encryption* (WEE) scheme for encrypting a matrix over the field $\mathbb{Z}_p$. The scheme has the *echelon transformation* property. We can apply a series of elementary row/column operations to transform an encrypted matrix under our WEE scheme into the row/column echelon form. The decrypted result matches the result of the corresponding operations performed on the original matrix. For security, our WEE scheme is shown to be entry irrecoverable for non-zero entries under the computational Diffie-Hellman assumption.
By using our WEE scheme, we propose five secure outsourcing protocols of Gaussian elimination, Gaussian-Jordan elimination, matrix determinant, linear system solver, and matrix inversion. Each of these protocols preserves data privacy for clients (data owners). Furthermore, the linear system solver and matrix inversion protocols provide a cheating-resistant mechanism to verify correctness of computation results. Our experimental result shows that our protocols gain efficiency significantly for an outsourcer. Our outsourcing protocol solves a linear system of $n = 1,000$ equations and $m = 1,000$ unknown variables about 472 times faster than a non-outsourced version. The efficiency gain is more substantial when (n, m) gets larger. For example, when $n = 10,000$ and $m = 10,000$, the protocol can solve it about $56,274$ times faster. Our protocols can also be easily implemented in parallel computation architecture to get more efficiency improvement.

Keywords. Secure outsourcing, data privacy, cloud computing, linear algebra, linear system

1 Introduction

Outsourcing large-scale computations for engineering and business is an important use of cloud computing. A resource-constrained outsourcer can move his computational task to a cloud server (CS), which holds massive computation resources. Two major concerns for an outsourcer are security and efficiency gain. The security concern originates from the fact that CSs are usually managed by commercial companies, which are outside the trust domain of outsourcers. An outsourcer wants to preserve data privacy when outsourcing computational tasks over sensitive information, e.g., personal health records and shopping history, etc. Furthermore, an outsourcer may need to verify CS's computing results, since CS may be lazy and just return incorrect answers without any computation. For efficiency, an outsourcer's computation time should be substantially less than the time of performing the original task on his own.

Many approaches on securely outsourced computation are based on symmetric-key settings [1, 2, 8, 26, 27]. An outsourcer encrypts his own data to CS. Only the outsourcer himself can decrypt and verify the returned results. In this paper, we consider a different outsourcing scenario, shown in Figure 1, which captures the public-key concept for secure outsourcing. Data analyzer (DA) and data owners are two different roles. DA publishes his public-keys for

* This work was supported by project MOST 101-2221-E-009-074-MY3, Taiwan.

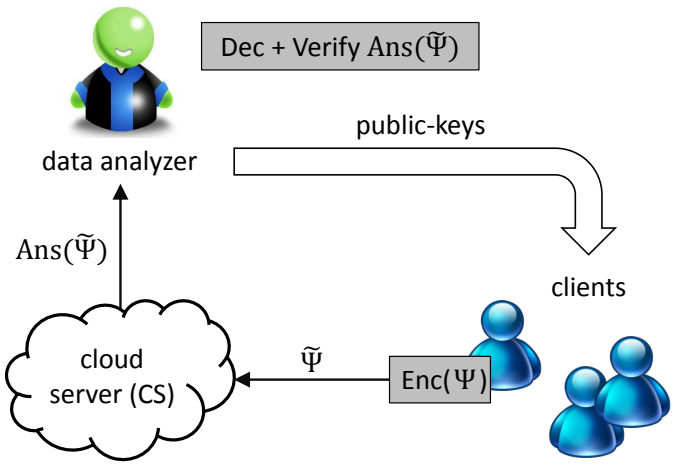


Fig. 1. Secure outsourcing data analytic computation scenario.

a data owner (client) to encrypt his data Ψ. The encrypted $\widetilde{\Psi}$ is sent to CS for computation. The result Ans($\widetilde{\Psi}$) is sent to DA for decryption and verification of correctness. This asymmetric setting is suitable in some applications, such as network flow characteristic finding and data-mining on distributed databases.

Contributions. In this paper, we propose outsourcing protocols of algebraic computation under the considered scenario. We focus on the *Gaussian elimination* (GE) and its related algebraic computations including solving a system of linear equations and computing inversion and determinant of a matrix. We propose a novel public-key *weave ElGamal encryption* (WEE) scheme for encrypting a matrix over $\mathbb{Z}_p$. Our WEE scheme has the *echelon transformation* (ET) property for a matrix over $\mathbb{Z}_p^{n\times m}$. The ET-property allows us to apply a series of elementary row/column operations (over $\mathbb{Z}_p$) to transform an encrypted matrix into an encrypted row/column echelon form. The decrypted result matches the result of the corresponding operations performed on the original matrix. For our WEE encryption scheme, an adversary cannot recover non-zero entries of an encrypted matrix if the computational Diffie-Hellman (CDH) problem is hard.

To outsource GE computation, a client uses the WEE scheme to encrypt his matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$ as $\widetilde{\mathbf{A}}$. CS performs echelon transformation on $\widetilde{\mathbf{A}}$ and gives the result matrix $\widetilde{\mathbf{A}}'$ to DA. Similarly, a client can outsource Gaussian-Jordan elimination (GJE), and matrix determinant computations without exposing input matrices. Furthermore, we propose two privacy-preserving and verifiable outsourcing protocols for solving linear systems and finding matrix inversion. DA can verify integrity and correctness of CS's returned results. Therefore, these two protocols are cheating-resistant against dishonest CSs.

Table 1 demonstrates the performance of our outsourcing protocols. In each protocol, an outsourcer (DA in our scenario) keeps at most $n + m$ private keys for decryption. Every protocol needs only two trips of communication: a client gives his encrypted matrix to CS and CS sends the computed result to the outsourcer. The decryption cost for an outsourcer is substantially less than that of performing the original computation himself.

We compare our outsourcing protocols with some recently related works in Table 1. The fully homomorphic encryption (FHE) approach allows a client to encrypt his data Ψ to CS.

Table 1. Performance of Our Algebraic Computation Outsourcing Protocols

Outsourcing protocol	Outsourcer private key	Computation		CS computation	Communication		Cheating-resistance	Computational model	
		Encryption	Decryption		#(trip)	#(element)		Client-server	Data analytic
GE	$n+m$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	$\mathcal{O}(n^2m)$	2	$2nm+2$	No	O	O
matrix determinant	1	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$\mathcal{O}(n^3)$	2	n^2+3	No	O	O
linear system solver	n	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	2	$2n^2+2n+2$	Yes	O	O
matrix inversion	n	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	2	$2n^2+n+2$	Yes	O	O
Other Approaches									
FHE approach on Ψ	$\mathcal{O}(1)$	$\mathcal{O}(\lvert\Psi\rvert)$	$\mathcal{O}(\lvert\text{Ans}(\Psi)\rvert)$	$\mathcal{O}(\Psi)$	2	$2\cdot\mathcal{O}(\lvert\Psi\rvert)$	Yes	O	O
linear system solver [2]	$\approx 2n$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	2	n^2+n	No	O	X
matrix inversion [27]	$\approx 2n$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	2	$2n^2$	Yes	O	X
matrix determine [26]	$\approx 2n$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	2	$2n^2$	Yes	O	X
linear system solver [8]	λn^2+n	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n^3)$	2	n^2+n	Yes	O	X

- The proposed GE/GJE outsourcing protocols can be applied to $n \times m$ rectangle matrices. The proposed linear system solver, matrix inversion, and matrix determinant outsourcing protocols are applied to $n \times n$ invertible matrices.

CS can do arbitrary computation on $\widetilde{\Psi}$ for DA. This approach is theoretically simple. However, a bottleneck occurs in implementing the existing FHE schemes [31]. The outsourcing protocols in [2, 8, 26, 27] use random echelon row and column operations over $\mathbb{R}$ to encrypt input matrices. However, they are for the symmetric-key setting, and not applicable in our scenario of public-key setting. Apparently, public-key computations are more expensive than symmetric-key ones. Thus, our protocols are less efficient than the protocols in [2, 8, 26, 27]. Nevertheless, our experiments show that our approach can help an outsourcer gain effective efficiency. For example, our outsourcing protocol solves a linear system of $n = 1,000$ equations and $m = 1,000$ unknown variables about 472 times faster than a non-outsourced version. The efficiency gain is more substantial when (n, m) gets larger. For example, when $n = 10,000$ and $m = 10,000$, the protocol can solve it about $56,274$ times faster.

The model of secure two-party (or multi-party) computation is similar to computation outsourcing. The parties cooperatively compute a specific function without exposing their input data. However, the model usually requires each client to participate the whole computation. It is not suitable in our considered computation scenario. Thus, we do not take these approaches into account in our comparison.

Related works. Securely outsourced computation has attracted considerable interest in the past. The breakthrough of Gentry's fully homomorphic encryption (FHE) scheme [18] provided a generic solution for an external agent to perform arbitrary computations on encrypted data. Many improvements were proposed for providing a better efficiency [6, 9, 34, 19]. Some generic solutions for verifiable outsourcing computations [10, 17] were proposed by using FHE schemes. Nevertheless, the performance of the existing FHE scheme are still far away from practicality [31].

In the past few years, many secure outsourcing protocols have been designed for algebraic computations [1, 2, 5, 8, 15, 24, 26, 27, 29]. Atallah et al. [2] designed some protocols for outsourcing scientific computations, such as matrix multiplication, matrix inversion, linear system solver, sorting, and string pattern matching. The general idea is to disguise input data before outsourcing them to CSs. Lei et al. [26, 27] proposed two secure outsourcing protocols for large matrix inversion and determinant computations with cheating-resistant mechanism. They also provided explicit protocol evaluation to show the efficiency gain of a client. A rigorous security proof for their protocols remains opened. Benjamin and Atallah [5] and Atallah and Frikken [1] proposed two private-preserving and cheating-resistant

outsourcing protocols for matrix multiplication computation. The former protocol [5] is built on the assumption of two non-colluding servers, and thus vulnerable to colluding attacks. The later protocol [1] uses one un-trusted server only. It is provably secure based on the use of Shamir's secret sharing scheme [33]. The computation cost for a client is $\mathcal{O}(t^2n^2)$, where n^2 is the size of an input matrix and t is the threshold in the secret sharing scheme.

In 2007, Kiltz et al. [24] proposed secure two-party protocols for various core problems in linear algebra, such as computing the rank of a matrix and solving a system of linear equations. The core of the protocols exploits the property of linearly recurrent sequences and their relation to the minimal polynomial of a matrix. The construction is provably-secure based on the use of additively homomorphic encryption schemes and Yao's garbled circuit protocol [35]. In their protocols, the overall communication complexity is roughly $\mathcal{O}(n^2)$. The round complexity is $polylog(n)$, which is better than $n^{0.275}$ of the communication efficient protocol [32]. Mohassel [29] proposed a non-interactive protocols with verification for outsourcing matrix multiplication. The construction uses a number of public-key encryption schemes with limited homomorphic properties. They also proposed outsourcing protocols for solving a system of linear equations and computing matrix inversion. These protocols have a constant round communication and $\mathcal{O}(n^2 \log n)$ computations cost for a client.

In 2012, Fiore and Gennaro [15] addressed the problem of public verification on outsourced computation for large polynomial evaluation and matrix multiplications. The result of an outsourced computation can be verified by a third party without secret keys.

There are some other kinds of works that are functionally and conceptually related to this research.

- Secure multi-party computation (SMPC). An SMPC protocol is for two (or more) parties to jointly evaluate a function while preserving the privacy of each party's input. Yao's garbled circuit [35] is the best known generic solution for SMPC. Many subsequent works [12, 23] made the computation increasingly practical even in the presence of malicious parties. Some works addressed the problem of balanced work load for each party [7, 21].
- Cooperative algebraic computation. This kind of protocol is for a set of parties (or databases) to cooperatively solve an algebraic problem. Each party holds a portion of input data and obtains a partial answer in the end of protocol execution. There were protocols for solving a system of linear equations [11, 13, 30, 32] and computing matrix multiplication [14, 22], etc.
- Server-aided computation. This kind of protocol is for a resource-constrained client to perform heavy cryptographic computations by using computing power of the server. There were protocols for RSA computation [3, 28] and modular exponentiation [20].

2 Preliminaries

In this section, we explain the notation in the paper and introduce the Gaussian elimination method.

Algorithm 1 Gaussian Elimination

Input: $\mathbf{A} = [a_{i,j}]_{n\times m} = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_n]^\top$
Output: a row echelon form of $\mathbf{A}$ and a permutation vector $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$
1: initialize a permutation vector $\pi \leftarrow (1, 2, \ldots, n)$;
2: **for** $k = 1$ **to** n **do**
3: find a nonzero $a_{z,k} \in \{a_{i,k} : k \le i \le n\}$;
4: **if** no such $a_{z,k}$ exists **then**
5: **error** "$\mathbf{A}$ is singular";
6: **end if**
7: $\pi_k \leftrightarrow \pi_z$;
8: $\mathbf{a}_k \leftrightarrow \mathbf{a}_z$;
9: **for** $i = k + 1$ **to** n **do**
10: $\mathbf{a}_i \leftarrow \mathbf{a}_i - (a_{i,k}/a_{k,k}) \cdot \mathbf{a}_k$;
11: **end for**
12: **end for**
13: **return** $(\mathbf{A}, \pi)$

2.1 Notations

A bold uppercase letter $\mathbf{A}_{n\times m} = [a_{i,j}]_{n\times m}$ is an $n \times m$ matrix, where $a_{i,j} \in \mathbb{Z}_p$ is the entry of the i-th row and j-th column. A bold lowercase letter $\mathbf{x}_{n\times 1} = [x_i]_{n\times 1}$ is an $n \times 1$ column matrix, where $x_i \in \mathbb{Z}_p$ is the i-th row entry. $\mathbf{x}_{n\times 1}$ can be either an n-dimension or an n-tuple vector $(x_1, x_2, \ldots, x_n)$. We simply use $\mathbf{A}$ and $\mathbf{x}$ to represent $\mathbf{A}_{n\times m}$ and $\mathbf{x}_{n\times 1}$ when their sizes are clearly understood. $\mathbf{A}^\top$ and $\mathbf{x}^\top$ are the transpose matrices of $\mathbf{A}$ and $\mathbf{x}$. Let $x \xleftarrow{\$} \mathcal{X}$ denote that x is randomly drawn from the set $\mathcal{X}$.

2.2 Gaussian Elimination (GE) Method

The Gaussian elimination (GE) method uses a sequence of elementary row operations to transform an $n \times m$ matrix $\mathbf{A} = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_n]^\top$ into a row echelon form matrix, whose bottom left corner is filled with zeros. This process is known as the *row echelon transformation*. The GE method can be performed over any field, e.g., $\mathbb{R}$ or $\mathbb{Z}_p$. There are three types of elementary row operations:

- *Row swapping*, denoted by $\mathbf{a}_\alpha \leftrightarrow \mathbf{a}_\beta$, swaps rows $\mathbf{a}_\alpha$ and $\mathbf{a}_\beta$.
- *Row multiplication*, denoted by $\mathbf{a}_\alpha \leftarrow \delta \cdot \mathbf{a}_\alpha$, multiplies a row $\mathbf{a}_\alpha$ with a non-zero scalar δ.
- *Row addition*, denoted by $\mathbf{a}_\beta \leftarrow \mathbf{a}_\beta + \delta \cdot \mathbf{a}_\alpha$, add one row $\mathbf{a}_\alpha$ multiplied with a non-zero scalar δ to another row $\mathbf{a}_\beta$.

The Gaussian elimination is shown in Algorithm 1. The pivoting process in lines 2-8 selects a nonzero entry from $\{a_{k,k}, a_{k+1,k}, \ldots, a_{n,k}\}$ as the pivot element. The main loop in lines 9-11 uses row addition operation $\mathbf{a}_i \leftarrow \mathbf{a}_i - \delta \cdot \mathbf{a}_k$ to eliminate the entries below $a_{i,k}$ (in the same column) into zeros. The algorithm needs $\mathcal{O}(n^2 m)$ arithmetic operations.

3 Our Weave ElGamal Encryption (WEE) Scheme

The scheme is a *multiple* ElGamal encryption scheme [4, 16, 25]. The key generation algorithm generates n key pairs $\langle s_i, S_i = g^{s_i}\rangle_{i=1}^n$ for rows and m key pairs $\langle t_j, T_j = g^{t_j}\rangle_{j=1}^m$ for columns.

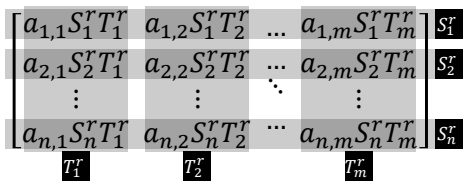


Fig. 2. The encryption idea of our WEE scheme.

For an $n \times m$ matrix $\mathbf{A} = [a_{i,j}]_{n\times m}$, a client picks a random $r \in \{1, 2, \ldots, p-2\}$ and encrypts $\mathbf{A}$ as

$$g^r, \widetilde{\mathbf{A}} = [a_{i,j} S_i^r T_j^r]_{n\times m}$$

The client uses S_i^r's and T_j^r's to encrypt $\mathbf{A}$ in a weaved form, which is shown in Fig. 2. The entries of $\widetilde{\mathbf{A}}$ have a homomorphism-like property, called the ET-property later, for elementary row operations in the GE method over $\mathbb{Z}_p$.

3.1 The Construction

Our WEE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ consists of key generation algorithm **KG**, encryption algorithm **Enc**, and decryption algorithm **Dec**.

- $\mathbf{KG}(\lambda, n, m) \rightarrow (\mathcal{PK}, \mathcal{SK})$**.** This algorithm takes as input a security parameter λ and integers (n, m), and does the following.
  - Generate a λ-bit secure prime $p = 2q + 1$ (q is also a prime) and a generator g for the cyclic group $\mathbb{Z}_p^*$.
  - Pick random integers $s_i, t_j \in [1, p-2]$ and compute $S_i = g^{s_i}$ and $T_j = g^{t_j}$, $1 \le i \le n$, $1 \le j \le m$.
  - The public and private key pair is $(\mathcal{PK}, \mathcal{SK})$, where $\mathcal{PK} = \{p, g, \langle S_i\rangle_{i=1}^n, \langle T_j\rangle_{j=1}^m\}$ and $\mathcal{SK} = \{\langle s_i\rangle_{i=1}^n, \langle t_j\rangle_{j=1}^m\}$.
- $\mathbf{Enc}(\mathcal{PK}, \mathbf{A}) \rightarrow (V, \widetilde{\mathbf{A}})$**.** This algorithm picks a random integer $r \in \{1, 2, \ldots, p-2\}$ and encrypts the input matrix $\mathbf{A} = [a_{i,j}]_{n\times m}$ as $V \leftarrow g^r, \widetilde{\mathbf{A}} \leftarrow [a_{i,j} S_i^r T_j^r]_{n\times m}$.
- $\mathbf{Dec}(\mathcal{SK}, V, \widetilde{\mathbf{A}}) \rightarrow \mathbf{A}$**.** This algorithm uses the private key $\mathcal{SK}$ to decrypt $\widetilde{\mathbf{A}} = [\widetilde{a}_{i,j}]_{n\times m}$ as $\mathbf{A} \leftarrow [\widetilde{a}_{i,j}/V^{s_i+t_j}]_{n\times m}$.

  Correctness. The decrypted result is correct since $a_{i,j} \leftarrow \widetilde{a}_{i,j}/V^{s_i+t_j} = a_{i,j} S_i^r T_j^r / (g^r)^{s_i} (g^r)^{t_j}$.

3.2 The Echelon Transformation (ET) Property

In this section, we give the definition of the echelon transformation property over a field $\mathbb{F}$ and show that our WEE scheme has such a property.

Definition 1 (ET-Property over Fields). *Let $\Phi : \mathbf{A} \rightarrow \widetilde{\mathbf{A}}$ be a matrix encryption scheme with entries in a field $\mathbb{F}$. We say that Φ has the echelon transformation (ET) property over field $\mathbb{F}$ if, after transforming $\widetilde{\mathbf{A}}$ into an echelon form $\widetilde{\mathbf{A}}'$ by a series of elementary row operations, the decrypted result of $\widetilde{\mathbf{A}}'$ matches the result of the corresponding operations performed on the original $\mathbf{A}$.*

Theorem 1. *Our WEE scheme Π has the ET-property over $\mathbb{Z}_p$.*

Proof. Let $\widetilde{\mathbf{A}} = [\widetilde{\mathbf{a}}_1 \ \cdots \ \widetilde{\mathbf{a}}_n]^\top$ be the encrypted $n \times m$ matrix $\mathbf{A}$ under Π. Let $\mathcal{PK}_i = \{p, g, S_i, \langle T_j \rangle_{j=1}^m\}$ be the public keys for encrypting the i-th row $\mathbf{a}_i$ and $\mathcal{PK}_{i,j} = \{p, g, S_i, T_j\}$ be the public key for encrypting $a_{i,j}$. The following checks correctness of each elementary operation.

- *Row swapping.* For $\widetilde{\mathbf{a}}_\alpha \leftrightarrow \widetilde{\mathbf{a}}_\beta$, the ciphertexts of two rows are unchanged.
- *Row multiplication.* For $\widetilde{\mathbf{a}}_\alpha \leftarrow \delta \cdot \widetilde{\mathbf{a}}_\alpha$,

$$\delta \cdot \mathbf{Enc}(\mathcal{PK}_\alpha, \mathbf{a}_\alpha) = \delta \cdot [a_{\alpha,j} S_\alpha^r T_j^r]_{1\times m} = [(\delta \cdot a_{\alpha,j}) S_\alpha^r T_j^r]_{1\times m} = \mathbf{Enc}(\mathcal{PK}_\alpha, \delta \cdot \mathbf{a}_\alpha)$$

- *Row addition.* For $\widetilde{\mathbf{a}}_\beta \leftarrow \widetilde{\mathbf{a}}_\beta + (\widetilde{a}_{\beta,\alpha}/\widetilde{a}_{\alpha,\alpha}) \cdot \widetilde{\mathbf{a}}_\alpha$,

$$\begin{aligned}
&\mathbf{Enc}(\mathcal{PK}_\beta, \mathbf{a}_\beta) + \frac{\mathbf{Enc}(\mathcal{PK}_{\beta,\alpha}, a_{\beta,\alpha})}{\mathbf{Enc}(\mathcal{PK}_{\alpha,\alpha}, a_{\alpha,\alpha})} \cdot \mathbf{Enc}(\mathcal{PK}_\alpha, \mathbf{a}_\alpha) \\
&= [a_{\beta,j} S_\beta^r T_j^r]_{1\times m} + \frac{a_{\beta,\alpha} S_\beta^r T_\alpha^r}{a_{\alpha,\alpha} S_\alpha^r T_\alpha^r} \cdot [a_{\alpha,j} S_\alpha^r T_j^r]_{1\times m} \\
&= [(a_{\beta,j} + \frac{a_{\beta,\alpha}}{a_{\alpha,\alpha}} \cdot a_{\alpha,j}) S_\beta^r T_j^r]_{1\times m} = \mathbf{Enc}(\mathcal{PK}_\beta, \mathbf{a}_\beta + \frac{a_{\beta,\alpha}}{a_{\alpha,\alpha}} \cdot \mathbf{a}_\alpha)
\end{aligned}$$

The decrypted results are the results of the corresponding operations on $\mathbf{A}$. □

3.3 Security Analysis

In this section, we show that our WEE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$ for encrypting a matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$ is *entry irrecoverable* for non-zero entries[1] if the CDH assumption holds. We consider the following *entry recovery* (ER) security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup phase. $\mathcal{C}$ runs $\mathbf{KG}(\lambda, n, m)$ to generate the public and private key pair $(\mathcal{PK}, \mathcal{SK})$ and gives $\mathcal{PK}$ to $\mathcal{A}$.

Challenge phase. $\mathcal{C}$ randomly picks an $n \times m$ matrix $\mathbf{A} = [a_{i,j}]_{n\times m}$, encrypts it as $\widetilde{\mathbf{A}}$, and sends $\widetilde{\mathbf{A}}$ to $\mathcal{A}$.

Guess phase. $\mathcal{A}$ outputs his guess $\overline{a}_{i,j}$ of an entry $a_{i,j}$.

The advantage of $\mathcal{A}$ winning the ER game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ER}} := \Pr[\overline{a}_{i,j} = a_{i,j} : \text{for an entry } a_{i,j}].$$

We say that Π is *entry irrecoverable* if no poly-time $\mathcal{A}$ can win the ER security game with a non-negligible $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ER}}$ (in λ).

[1] In our WEE encryption scheme, we do not consider the privacy for $\mathbf{A}$'s zero entries. If $a_{i,j}$ is zero, the encrypted $\widetilde{a}_{i,j}$ is also zero. In the original ElGamal encryption scheme, the plaintext set $\mathcal{M} = \mathbb{Z}_p^*$. To ensure validity of the GE method, we need to extend the group $\mathbb{Z}_p^*$ to the field $\mathbb{Z}_p = \mathbb{Z}_p^* \cup \{0\}$. However, the ciphertext of zero message is zero in most ElGamal-like encryption schemes. It is interesting to find a secure public-key encryption scheme that supports GE computations over $\mathbb{Z}_p$, or a field $\mathbb{F}$, and hides zero entries.

Definition 2 (CDH Assumption over G). *Given g^u, $g^v \in G = \langle g \rangle$, computing g^{uv} is intractable. That is, the advantage*

$$\mathsf{Adv}_{\mathcal{B}}^{CDH} := \Pr[\mathcal{B}(g, g^u, g^v) = g^{uv}]$$

is negligible for any poly-time algorithm $\mathcal{B}$.

Theorem 2. *Our WEE scheme Π is entry irrecoverable for non-zero entries if the CDH assumption holds.*

Proof. We construct an algorithm $\mathcal{B}_{\mathrm{CDH}}$ attacking the CDH assumption in Algorithm 2 by using an adversary $\mathcal{A}$ attacking our WEE scheme Π under the ER security game.

Algorithm 2 $\mathcal{B}_{\mathrm{CDH}}$

Input: (n, m), (p, g, g^u, g^v), $(\mathcal{A}, \Pi)$
Output: g^{uv}
1: **for** $i = 1$ **to** n **do**
2: pick $\alpha_i \xleftarrow{\$} \{1, 2, \ldots, p-2\}$ and compute $S_i \leftarrow (g^u)^{\alpha_i}$;
3: **end for**
4: **for** $j = 1$ **to** m **do**
5: pick $\beta_j \xleftarrow{\$} \{1, 2, \ldots, p-2\}$ and compute $T_j \leftarrow (g^u)^{\beta_j}$;
6: **end for**
7: set $\mathcal{PK} \leftarrow \{p, g, \langle S_i \rangle_{i=1}^n, \langle T_j \rangle_{j=1}^m\}$;
8: set $V \leftarrow g^v$ and pick $\widetilde{\mathbf{A}} = [\widetilde{a}_{i,j}] \xleftarrow{\$} \mathbb{Z}_p^{n \times m}$;
9: $\overline{a}_{i,j} \leftarrow \mathcal{A}(V, \widetilde{\mathbf{A}})$;
10: **return** $\widetilde{a}_{i,j} / \overline{a}_{i,j}$

$\mathcal{B}_{\mathrm{CDH}}$ takes the parameters (n, m), CDH problem instantiation (p, g, g^u, g^v), and algorithms $(\mathcal{A}, \Pi)$ as inputs. In the setup phase of lines 1-6, $\mathcal{B}_{\mathrm{CDH}}$ simulates the system public key $\mathcal{PK} = \{p, g, \langle S_i \rangle_{i=1}^n, \langle T_j \rangle_{j=1}^m\}$ for $\mathcal{A}$ by using g^u. The private keys $\langle s_i \rangle_{i=1}^n$ and $\langle t_j \rangle_{j=1}^m$ are implicitly set as (unknown) $\langle \alpha_i u \rangle_{i=1}^n$ and $\langle \beta_j u \rangle_{j=1}^m$, respectively. In the challenge phase of lines 7-8, $\mathcal{B}_{\mathrm{CDH}}$ simulates the challenged $(V, \widetilde{\mathbf{A}})$ for $\mathcal{A}$ by setting $V = g^v$. Each entry $\widetilde{a}_{i,j}$ is implicitly set as $a_{i,j} g^{uv}$ with an (unknown) $a_{i,j} \in \mathbb{Z}_p$. In the guess phase of lines 9-10, if $\mathcal{A}$ outputs his guess $\overline{a}_{i,j}$ for $a_{i,j}$ correctly, $\mathcal{B}_{\mathrm{CDH}}$ computes $g^{uv} \leftarrow \widetilde{a}_{i,j} / \overline{a}_{i,j}$ correctly. Therefore, the advantage of $\mathcal{B}_{\mathrm{CDH}}$ breaking the CDH assumption is:

$$\mathsf{Adv}_{\mathcal{B}_{\mathrm{CDH}}}^{\mathrm{CDH}} = \Pr[\mathcal{B}_{\mathrm{CDH}}(g, g^u, g^v) = g^{uv}] = \Pr[\overline{a}_{i,j} = a_{i,j} : \overline{a}_{i,j} \leftarrow \mathcal{A}(V, \widetilde{\mathbf{A}})] = \mathsf{Adv}_{\mathcal{A}}^{\mathrm{ER}}.$$

Thus, if there exists a poly-time adversary $\mathcal{A}$ who can win the ER security game against our WEE scheme Π with a non-negligible advantage, $\mathcal{A}$ can be used to break the CDH assumption. □

4 Our Outsourcing Protocols for Algebraic Computations

By using our WEE scheme $\Pi = (\mathbf{KG}, \mathbf{Enc}, \mathbf{Dec})$, we can construct our outsourcing protocols for GE, GJE, matrix determinant, linear system solver, and matrix inversion computations.

System model. We consider the computation architecture as shown in Figure 1. A resource-constrained DA wants to outsource his data analysis task on a clients' data to CS, which is assumed to be honest-but-curious (semi-honest). CS is honest in his computational tasks, but will try his best to retrieve the inputted client's data and outputted result. Furthermore, in some of our protocols, we also consider CS as an "active" attacker. For example, CS may send an invalid result to DA. The communication channels among CS, clients, and DA are assumed to be secure. The transmitted information can be authenticated by the receiver. We can use the secure communication protocols, such as TLS, to establish the channels. In our system, we assume that DA's computation and communication capabilities are the same as the resourced-constrained clients. DA needs CS's help to accomplish his expensive computation on data analysis. In addition, we assume that the private-keys of DA and clients would not be leaked to an unauthorized party.

An outsourcing protocol consists of the following phases:

- **Initialization** phase: DA sets up the system, generates his public- and private-key pair, and publishes the public keys.
- **Outsourcing** phase: A client submits his encrypted data to CS.
- **Acquirement** phase: DA computes the final result from the CS's computed result.
- (Optional) **Verification** phase: DA checks correctness of CS's computed result.

4.1 The GE/GJE Outsourcing Protocol

- **Initialization phase.** DA chooses parameters (λ, n, m), generates $(\mathcal{PK}, \mathcal{SK}) \leftarrow \mathbf{KG}(\lambda, n, m)$ and publishes $\mathcal{PK}$.
- **Outsourcing phase.** A client encrypts his input matrix $\mathbf{A}$ as $(V, \widetilde{\mathbf{A}}) \leftarrow \mathbf{Enc}(\mathcal{PK}, \mathbf{A})$ and submits it to CS. CS performs the GE algorithm over $\mathbb{Z}_p$ to obtain $(\widetilde{\mathbf{A}}', \pi) \leftarrow \mathbf{GE}(\widetilde{\mathbf{A}})$ and gives $(V, \widetilde{\mathbf{A}}', \pi)$ to DA.
- **Acquirement phase.** DA applies π on $\langle s_i \rangle_{i=1}^n$ and computes the answer $\mathbf{A} \leftarrow \mathbf{Dec}(\mathcal{SK}, V, \widetilde{\mathbf{A}}')$.

Efficiency. In this protocol, DA needs to store $|\mathcal{SK}| = n+m$ private keys. For encryption, a client needs $n+m+1$ modular exponentiations to compute $V = g^r$, $\langle S_i^r \rangle_{i=1}^n$, and $\langle T_j^r \rangle_{j=1}^m$, and $2nm$ modular multiplications to compute $\widetilde{\mathbf{A}} = [a_{i,j} S_i^r T_j^r]_{n \times m}$. For decryption, the DA needs $nm - (\min\{m-1, n-1\})^2/2$ operations of modular addition, exponentiation, inverse, and multiplication to compute $a_{i,j} \leftarrow \widetilde{a}_{i,j} / V^{s_i + t_j}$ for the upper triangular part of $\mathbf{A}'$. CS performs $\mathcal{O}(n^2 m)$ arithmetic operations in the GE algorithm. For communication, the client uploads $1+nm$ group elements to CS and CS gives $1+nm$ (at most) group elements to DA.

Remark. The Gaussian-Jordan elimination (GJE) method is a variant of the GE method. It transforms a matrix into its reduced echelon form via a series of elementary row/column operations. The GJE outsourcing protocol is similar to the GE outsourcing protocol.

4.2 The Matrix Determinant Outsourcing Protocol

The following shows our matrix determinant outsourcing protocol for an invertible matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times n}$. CS first uses the GE method, without applying row multiplication operations, to transform the encrypted matrix $\widetilde{\mathbf{A}}$ into its echelon form $\widetilde{\mathbf{A}}'$. CS then computes the encrypted determinant value $\det(\widetilde{\mathbf{A}}) = (-1)^{|\pi|} \cdot \prod_i \widetilde{a}'_{i,i}$ for DA, where $|\pi|$ is the number of row swappings in performing the GE algorithm.

– **Initialization phase.** DA chooses parameters (λ, n), generates $(\mathcal{PK}, \mathcal{SK}) \leftarrow \mathbf{KG}(\lambda, n, n)$, computes $K = \sum_{i=1}^{n}(s_i + t_i)$, and publishes $\mathcal{PK}$.
– **Outsourcing phase.** A client encrypts his input matrix $\mathbf{A}$ as $(V, \widetilde{\mathbf{A}}) \leftarrow \mathbf{Enc}(\mathcal{PK}, \mathbf{A})$ and submits it to CS. CS performs the GE algorithm over $\mathbb{Z}_p$, with applying row swapping and addition operations only, to obtain $(\widetilde{\mathbf{A}}', \pi) \leftarrow \mathbf{GE}(\widetilde{\mathbf{A}})$. CS then computes $\det(\widetilde{\mathbf{A}}) = (-1)^{|\pi|} \cdot \prod_i \widetilde{a}'_{i,i}$ and gives $(V, \det(\widetilde{\mathbf{A}}))$ to DA.
– **Acquirement phase.** DA computes the answer

$$\det(\mathbf{A}) = \det(\widetilde{\mathbf{A}})/V^K. \tag{1}$$

Correctness. Due to the ET-property of Π, the diagonal entries of $\widetilde{\mathbf{A}}'$ are $a'_{1,1}S_1^r T_1^r$, $a'_{2,2}S_2^r T_2^r$, …, $a'_{n,n}S_n^r T_n^r$. The encrypted determinant value is

$$\det(\widetilde{\mathbf{A}}) = (-1)^{|\pi|} \cdot \prod_{i=1}^{n} a'_{i,i} S_i^r T_i^r = (-1)^{|\pi|} \cdot \prod_{i=1}^{n} a'_{i,i} \cdot g^{r\sum_{i=1}^{n}(s_i+t_i)}.$$

Thus, DA can compute the answer by Equation (1).

Efficiency. In this protocol, DA needs to store one private key K only. The matrix encryption cost is the same as that in our GE/GJE outsourcing protocol for $n = m$. For decryption, DA needs only one modular exponentiation, inverse, and multiplications to obtain the answer $\det(\mathbf{A})$. CS performs $\mathcal{O}(n^3)$ arithmetic operations in the GE algorithm. For communication, the client uploads $1 + n^2$ group elements to CS and CS gives 2 group elements to DA.

4.3 The Linear System Solver Outsourcing Protocol

The following shows our linear system solver outsourcing protocol for a system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ invertible matrix. CS uses the GE method to transform the linear system's augmented matrix $\mathbf{G} = [\mathbf{A}|\mathbf{b}]$ into its row echelon form $\mathbf{G}'$, and solves it by using the back substitution method.

– **Initialization phase.** DA chooses parameters (λ, n), generates $(\mathcal{PK}, \mathcal{SK}) \leftarrow \mathbf{KG}(\lambda, n, n+1)$, and publishes $\mathcal{PK}$.
– **Outsourcing phase.** A client encrypts the system's augmented matrix $\mathbf{G} = [\mathbf{A}|\mathbf{b}]$ as $(V, \widetilde{\mathbf{G}}) \leftarrow \mathbf{Enc}(\mathcal{PK}, \mathbf{G})$ and submits it to CS. CS performs the GE algorithm over $\mathbb{Z}_p$ to obtain $(\widetilde{\mathbf{G}}', \pi) \leftarrow \mathbf{GE}(\widetilde{\mathbf{G}})$, where $\widetilde{\mathbf{G}}' = [\widetilde{\mathbf{A}}'|\widetilde{\mathbf{b}}']$. Then, CS solves $\widetilde{\mathbf{A}}'\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}}'$ for $\widetilde{\mathbf{x}}$ via the back substitution method and gives $(V, \widetilde{\mathbf{x}})$ to DA.
– **Verification phase.** DA checks whether $\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}}$ holds. If not, he rejects the computed result $\widetilde{\mathbf{x}}$ and accepts $\widetilde{\mathbf{x}}$ otherwise.
– **Acquirement phase.** If DA accepts $\widetilde{\mathbf{x}}$, he computes

$$\mathbf{x} \leftarrow [\widetilde{x}_i/(V^{t_{n+1}-t_i})]_{n\times 1} \tag{2}$$

Correctness. We have $\widetilde{\mathbf{G}}' = [\widetilde{a}'_{i,j}|\widetilde{b}'_{i,n+1}]$, where $\widetilde{\mathbf{G}}'$'s upper $n \times (n+1)$ part consists of nonzero rows. CS computes the solution $\widetilde{x}_n$, $\widetilde{x}_{n-1}$, …, $\widetilde{x}_1$ by the back substitution method

as follows.

$$\begin{aligned}
\widetilde{x}_n &= \frac{\widetilde{b}'_{n,n+1}}{\widetilde{a}'_{n,n}} = \frac{b'_{n,n+1}}{a'_{n,n}} \cdot \frac{T^r_{n+1}}{T^r_n} \\
\widetilde{x}_{n-1} &= \frac{\widetilde{b}'_{n-1,n+1} - \widetilde{a}'_{n-1,n} \cdot \widetilde{x}_n}{\widetilde{a}'_{n-1,n-1}} = \frac{b'_{n-1,n+1} S^r_{n-1} T^r_{n+1} - a'_{n-1,n} S^r_{n-1} T^r_n \cdot x_n \frac{T^r_{n+1}}{T^r_n}}{a'_{n-1,n-1} S^r_{n-1} T^r_{n-1}} \\
&= \frac{b'_{n-1,n+1} - a'_{n-1,n} \cdot x_n}{a'_{n-1,n-1}} \cdot \frac{T^r_{n+1}}{T^r_{n-1}} \\
&\vdots \\
\widetilde{x}_1 &= \frac{\widetilde{b}'_{1,n+1} - \sum_{\ell=2}^{n} \widetilde{a}'_{1,\ell} \cdot \widetilde{x}_\ell}{\widetilde{a}'_{1,1}} = \frac{b'_{1,n+1} S^r_1 T^r_{n+1} - \sum_{\ell=2}^{n} a'_{1,\ell} S^r_1 T^r_\ell \cdot x_\ell \frac{T^r_{n+1}}{T^r_\ell}}{a'_{1,1} S^r_1 T^r_1} \\
&= \frac{b'_{1,n+1} - \sum_{\ell=2}^{n} a'_{1,\ell} \cdot x_\ell}{a'_{1,1}} \cdot \frac{T^r_{n+1}}{T^r_1}
\end{aligned}$$

Thus, the client can compute the answer $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ by Equation (2).

Efficiency. In this protocol, DA needs to store n private keys. The encryption cost is the same as that in our GE/GJE/matrix determinant outsourcing protocol. DA needs only n operations of modular addition, exponentiation, inverse, and multiplication to compute the answer $\mathbf{x}$. For verification, DA needs n^2 operations of modular multiplications and $n(n-1)$ operations of modular addition. CS performs $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ arithmetic operations in the GE and back substitution methods, respectively. For communication, the client uploads $n^2 + n + 1$ group elements to CS and CS gives $n^2 + n + 1$ group elements to the DA.

4.4 The Matrix Inversion Outsourcing Protocol

The following shows our matrix inversion outsourcing protocol for an invertible matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times n}$. CS uses the GJE method to transform the matrix $\widetilde{\mathbf{G}} = [\widetilde{\mathbf{A}}|\widetilde{\mathbf{I}}]$ into its reduced row echelon form $\mathbf{G}' = [\mathbf{I}|\widetilde{\mathbf{B}}]$. $\widetilde{\mathbf{B}}$ is the encrypted $\mathbf{A}^{-1}$.

- **Initialization phase.** DA chooses parameters (λ, n), generates $(\mathcal{PK}, \mathcal{SK}) \leftarrow \mathbf{KG}(\lambda, n, 2n)$, and publishes $\mathcal{PK}$.
- **Outsourcing phase.** A client encrypts the matrix $\mathbf{G} = [\mathbf{A}|\mathbf{I}]$ as $(V, \widetilde{\mathbf{G}}) \leftarrow \mathbf{Enc}(\mathcal{PK}, \mathbf{G})$ ans submits it to CS. CS performs the GJE algorithm over $\mathbb{Z}_p$ to obtain $\widetilde{\mathbf{G}}' = [\mathbf{I}|\widetilde{\mathbf{B}}] \leftarrow \mathbf{GJE}(\widetilde{\mathbf{G}})$ and gives $(V, \widetilde{\mathbf{B}})$ to the DA.
- **Acquirement phase.** DA obtains $\widetilde{\mathbf{B}} = [\widetilde{b}_{i,n+j}]_{n \times n}$ and computes

$$\mathbf{B} = [\widetilde{b}_{i,j} / V^{t_{n+j} - t_i}]_{n \times n} \tag{3}$$

- **Verification phase.** DA performs the following check steps ℓ times:
  - Pick a random nonzero vector $\mathbf{r} \in \mathbb{Z}_p^{m \times 1}$.
  - Compute $\mathbf{c} = \mathbf{B}(\mathbf{A}\mathbf{r}) - \mathbf{I}\mathbf{r}$.
  - If $\mathbf{c} \neq \mathbf{0}$, DA rejects $\mathbf{B}$ and abort.

  If $\mathbf{B}$ passes the above check steps, DA accepts the answer $\mathbf{A}^{-1} = \mathbf{B}$.

Correctness. In the outsourcing phase, the transformed matrix $\widetilde{\mathbf{B}}$ is based on the GJE algorithm. Thus, the correctness of Equation (3) can be checked easily due to the ET-property of our WEE scheme.

Efficiency. In this protocol, DA needs to store $|\{t_{n+1}, \ldots, t_{2n}\}| = n$ private keys. For encryption, a client needs $3n + 1$ modular exponentiations to compute $V = g^r$, $\langle S_i^r \rangle_{i=1}^n$, and $\langle T_j^r \rangle_{j=1}^{2n}$, and $2n^2 + n$ modular multiplications to compute $\widetilde{\mathbf{G}}$. For decryption, DA needs n^2 operations of modular addition, exponentiation, inverse, and multiplication to compute $\mathbf{B}$. For verification, DA needs $\mathcal{O}(\ell \cdot n^2)$ arithmetic operations. CS performs $\mathcal{O}(n^3)$ arithmetic operations in the GJE algorithm. For communication, the client uploads $n^2 + n + 1$ group elements to CS and CS gives $n^2 + 1$ group elements to DA.

4.5 Security of Our Outsourcing Protocols

Base on our system model and assumptions (in the beginning of Section 4), we give a security analysis for each of our outsourcing protocols as follows. Recall that the main goal is to preserve a client's input and output data privacy against CS. An optional goal is to ensure correctness of CS's computed results by a verification mechanism.

Input/Output data privacy. In our GE/GJE and matrix determinant protocols, the privacy of a client's original $\mathbf{A}$ is preserved by our WEE scheme. In the outsourcing phase, CS cannot decrypt any nonzero entry $\widetilde{a}''_{i,j}$ of an intermediate matrix $\widetilde{\mathbf{A}}''$ in the process of transforming $\widetilde{\mathbf{A}}$ to $\widetilde{\mathbf{A}}'$. The reason is as follows. By the ET-property of our WEE scheme, we have that $\widetilde{a}''_{i,j} = a''_{i,j} S_i^r T_j^r$. If CS can decrypt $\widetilde{a}''_{i,j}$ as $a''_{i,j}$, he can compute $a'''_{i,j} = \widetilde{a}_{i,j}/(\widetilde{a}''_{i,j}/a''_{i,j})$ and obtain $\pi^{-1}(a'''_{i,j})$ in $\mathbf{A}$. It contradicts the security property of our WEE scheme. Thus, our GE/GJE and matrix determinant protocols preserve a client's input data privacy. For output data privacy, in our GE/GJE protocol, the result $\widetilde{\mathbf{A}}'$ is also an encryption of our WEE scheme. In our matrix determinant protocol, since CS does not have K (or $\mathcal{SK}$), CS cannot decrypt $\det(\widetilde{\mathbf{A}})$ due to the hardness of the CDH assumption.

Similarly, our linear system solver protocol preserves a client's input data privacy against CS. The linear system $(\mathbf{A}, \mathbf{b})$ is encrypted by our WEE scheme. Our linear system solver protocol preserves output data privacy as well. In the outsourcing phase, CS obtains $\widetilde{\mathbf{x}} = (x_1(T_{n+1}^r/T_1^r), \ldots, x_n(T_{n+1}^r/T_n^r))$. If CS can decrypt $\widetilde{x}_i = x_i(T_{n+1}^r/T_i^r)$ to obtain x_i, he can compute $b_i''' = \widetilde{b}'/(\widetilde{x}_i/x_i)$ and obtain $\pi^{-1}(b_i''')$ in $\mathbf{b}$. It leads to a contradiction of the security of our WEE scheme. Thus, the output privacy in our linear system solver protocol is preserved.

In our matrix inversion protocol, a client's original $\mathbf{A}$ is encrypted as $\widetilde{\mathbf{G}} = [\widetilde{\mathbf{A}}|\widetilde{\mathbf{I}}]$. The privacy of $\mathbf{A}$ is ensured as the previous protocols. For output privacy, the privacy of the result matrix $\mathbf{B}$ is preserved. Since $\widetilde{\mathbf{B}} = [b_{i,j}(T_i^r)^{-1} T_{n+j}^r]$ is an encryption of WEE scheme by regarding $(T_i^r)^{-1}$ as S_i^r and T_{n+j}^r as T_j in our original scheme description.

Output data verification. In our linear system solver protocol and matrix inversion protocols, DA can verify correctness of CS's computed results. In the linear system solver protocol, DA checks whether $\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}}$ to verify CS's computation. If CS is honest in computation, DA has an $\widetilde{\mathbf{x}}$ that satisfies $\widetilde{\mathbf{A}}\widetilde{\mathbf{x}} = \widetilde{\mathbf{b}}$. If CS is dishonest in computation, the probability that CS outputs a correct $\widetilde{\mathbf{x}}$ to pass DA's verification is $(\frac{1}{p})^n = (\frac{1}{2^\lambda})^n$, which is negligible in the security parameter λ.

In the verification procedure of our matrix inversion protocol, if CS is honest in computation, DA has $\mathbf{B} = \mathbf{A}^{-1}$ and $\mathbf{BA} = \mathbf{I}$. If CS is dishonest in computation, DA accepts $\mathbf{B}$

with probability $\leq 1/2^{\ell}$, which is negligible in ℓ. The reason is as follows. Let $\mathbf{Z} = \mathbf{BA} - \mathbf{I}$. If $\mathbf{B} \neq \mathbf{A}^{-1}$, at least one entry of $\mathbf{Z}$ is nonzero, say, $z_{i,j} \neq 0$. Let $c_i = \sum_{k=1}^{n} d_{i,k} r_k = d_{i,j} r_j + x$. We have

$$\begin{aligned}
\Pr[c_i = 0] &= \Pr[x = 0] \Pr[c_i = 0 | x = 0] + \Pr[x \neq 0] \Pr[c_i = 0 | x \neq 0] \\
&\leq \Pr[x = 0] \Pr[r_j = 0] + \Pr[x \neq 0] \Pr[r_j = 1] \\
&= \frac{1}{2}(\Pr[x = 0] + \Pr[x \neq 0]) = \frac{1}{2}.
\end{aligned}$$

The probability of verification failure in each iteration is $\Pr[\mathbf{c} = \mathbf{0}] \leq \Pr[c_i = 0] \leq \frac{1}{2}$. Therefore, after ℓ verification steps, the total probability p_f of verification failure is $p_f \leq (\Pr[\mathbf{c} = \mathbf{0}])^{\ell} \leq \frac{1}{2^{\ell}}$.

5 Performance Evaluation

We implement our linear system solver outsourcing protocol and evaluate its efficiency. In the first experiment, we demonstrate the essential gain of efficiency for DA, the outsourcer. The second experiment demonstrates the computation time for CS to solve linear systems in a parallel computing environment. We set the security parameter $\lambda = 1,024$, i.e., p is a $1,024$-bit prime. Our randomly generated linear system (has a unique solution) contains n equations and n unknown variables over the finite field $\mathbb{Z}_p$.

All experiments are run on an HP ProLiant DL165 G7 machine, which has an AMD Opteron 6128 2.0 GHz octa-core processor with four 8 GB unregistered DDR3-1333 memory modules. The machine runs Debian 7.6 AMD64 version with 128 GB swap space. The machine has a software RAID-0 device of 1 TB Ext4 partition for system usage and a software RAID-0 device of 4 TB Ext4 partition for experiment usage. We use POSIX threads for parallel programming and GNU multiple precision arithmetic library version 6.0.0 for big numbers.

5.1 Experimental Results

The numerical data of our first experiment are shown in Table 2. In this experiment, DA and client use one CPU core and CS uses eight CPU cores. In the baseline evaluation, T_{DA} and T_{CS} are the computation time of DA and CS to solve a linear system, respectively. In our protocol, T_{Enc} is the computation time of a client to encrypt his matrix and T_{Dec} is the computation time of DA to decrypt the final answer. The efficiency gain α is define as

$$\alpha := \frac{T_{\text{DA}}}{T_{\text{Dec}}}$$

We can see that α is greater than 1 for our chosen n's. Thus, it is worth for DA to outsource the computation by using our protocol.α is more substantial when n gets larger, such as, $\alpha \approx 472$ when $n = 1,000$ and $\alpha \approx 56274$ when $n = 10,000$. Figure 3(a) shows the efficiency gain α of selective n's.

Table 3 shows numerical data of our second experiment. T_k is the computation time for

Table 2. Experiment Result of Our Linear System Solver Outsourcing Protocol

	Matrix size		Baseline		Our protocol		Efficiency gain
n	$\lvert[\mathbf{A}\lvert\mathbf{b}]_{n\times(n+1)}\rvert$	$\lvert\mathbf{x}_{n\times 1}\rvert$	T_{DA}	T_{CS}	T_{Enc} (client)	T_{Dec} (DA)	$\alpha = T_{\text{DA}}/T_{\text{Dec}}$
50	331 KB	6.48 KB	69.83 ms	65.33 ms	103.78 ms	55.58 ms	1.26×
60	475 KB	7.78 KB	113.72 ms	92.13 ms	124.63 ms	64.93 ms	1.75×
70	645 KB	9.08 KB	177.35 ms	126.16 ms	146.50 ms	74.53 ms	2.38×
80	841 KB	10.37 KB	261.72 ms	172.41 ms	169.06 ms	84.21 ms	3.11×
90	1.04 MB	11.67 KB	369.19 ms	214.65 ms	194.68 ms	94.64 ms	3.90×
100	1.28 MB	12.96 KB	503.79 ms	274.36 ms	217.85 ms	102.37 ms	4.92×
110	1.55 MB	14.26 KB	679.08 ms	343.75 ms	241.46 ms	110.78 ms	6.13×
120	1.84 MB	15.56 KB	863.50 ms	411.82 ms	267.69 ms	118.83 ms	7.27×
130	2.16 MB	16.85 KB	1.10 sec	497.51 ms	293.34 ms	128.13 ms	8.59×
140	2.50 MB	18.16 KB	1.37 sec	526.14 ms	321.94 ms	139.08 ms	9.85×
150	2.87 MB	19.44 KB	1.68 sec	581.09 ms	347.89 ms	145.40 ms	11.55×
1000	127 MB	130 KB	8.33 min	1.30 min	5.21 sec	1.06 sec	471.51×
2000	507 MB	260 KB	1.11 hr	9.77 min	18.13 sec	1.95 sec	2049.23×
3000	1.11 GB	389 KB	3.77 hr	32.20 min	37.33 sec	2.84 sec	4778.87×
4000	1.98 GB	519 KB	8.88 hr	1.26 hr	1.03 min	3.97 sec	8052.39×
5000	3.09 GB	649 KB	17.46 hr	2.45 hr	1.54 min	4.76 sec	13205.04×
6000	4.65 GB	778 KB	1.26 day	4.25 hr	2.18 min	5.52 sec	19721.74×
7000	6.06 GB	908 KB	2.01 day	6.86 hr	2.94 min	6.62 sec	26233.23×
8000	7.91 GB	1.01 MB	3.02 day	10.38 hr	3.88 min	7.35 sec	35500.41×
9000	10.01 GB	1.14 MB	4.30 day	14.98 hr	4.91 min	8.21 sec	45252.13×
10000	12.36 GB	1.27 MB	5.94 day	20.72 hr	6.12 min	9.12 sec	56273.68×

the CS to solve a linear system by using k CPU cores in parallel, $k = 2, 4, 8$. The speedup ratio β_k is define as

$$\beta_k = \frac{T_1}{T_k}$$

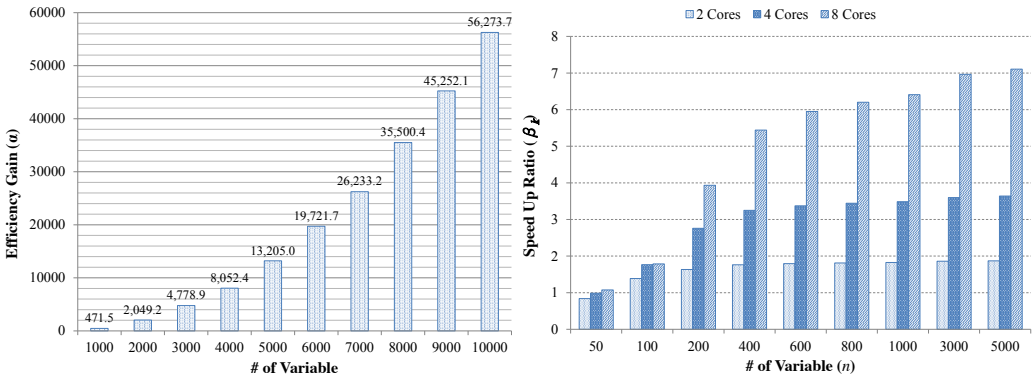
It represents the speedup ratio of using k CPU cores in parallel for computation. Table 3 shows that multiple CPU cores are better used when n gets larger. For example, if the CS uses 8 CPU cores, $\beta_8 \approx 5.78$ when $n = 500$ and $\beta_8 \approx 7.11$ when $n = 5,000$. Figure 3(b) shows the trend of the speedup ratios β_2, β_4, and β_8 for different numbers of CPU cores and variables.

6 Conclusion

In this paper, we propose a novel WEE scheme for encrypting a matrix over $\mathbb{Z}_p$. The scheme has the nice ET-property over $\mathbb{Z}_p$. Based on our WEE scheme, we propose five privacy-preserving and efficient outsourcing protocols for GE, GJE, matrix determinant, linear system solver, and matrix inversion. The linear system solver and matrix inversion protocols have cheating-resistant mechanisms. The experimental results show that our protocols have significant efficiency gain for a client. In practice, our works could be used in privacy-preserving cloud computing, statistical data analysis, machine learning tasks, etc.

The directions for our future works include the following.

- To find an encoding method from $\mathbb{R}$ to $\mathbb{Z}_p$ for our outsourcing protocols. In most applications, the linear algebraic computations are on real number set $\mathbb{R}$.
- To design cheating-resistant mechanisms for all of our outsourcing protocols. Now we only have cheating-resistant mechanisms for the linear system solver and matrix inversion



(a) The efficiency gain of a DA (b) The speedup ratios β_2, β_4, and β_8

Fig. 3. Our experimental results

Table 3. Experiment Result of Solving Linear System in Parallel

	1 Core	2 Cores	speedup	4 Cores	speedup	8 Cores	speedup
n	T_1	T_2	T_1/T_2	T_4	T_1/T_4	T_8	T_1/T_8
50	73.89 ms	88.01 ms	0.84×	75.06 ms	0.98×	68.78 ms	1.07×
60	121.24 ms	106.61 ms	1.14×	115.09 ms	1.05×	94.72 ms	1.28×
70	184.64 ms	153.80 ms	1.21×	161.29 ms	1.16×	128.62 ms	1.45×
80	264.85 ms	208.14 ms	1.27×	201.02 ms	1.32×	168.85 ms	1.57×
90	369.31 ms	277.37 ms	1.33×	254.84 ms	1.45×	221.66 ms	1.67×
100	503.78 ms	364.60 ms	1.39×	285.52 ms	1.76×	282.17 ms	1.79×
200	4.02 sec	2.46 sec	1.63×	1.46 sec	2.76×	1.02 sec	3.93×
300	13.58 sec	7.90 sec	1.72×	4.60 sec	3.09×	2.70 sec	5.03×
400	32.11 sec	18.23 sec	1.76×	9.88 sec	3.25×	5.90 sec	5.44×
500	1.04 min	35.38 sec	1.77×	18.85 sec	3.32×	10.83 sec	5.78×
600	1.80 min	1.00 min	1.79×	32.02 sec	3.37×	18.15 sec	5.95×
700	4.66 min	1.59 min	1.80×	50.09 sec	3.43×	28.37 sec	6.05×
800	4.27 min	2.35 min	1.81×	1.24 min	3.44×	41.26 sec	6.20×
900	6.06 min	3.33 min	1.82×	1.75 min	3.46×	58.38 sec	6.22×
1000	8.32 min	4.56 min	1.83×	2.39 min	3.48×	1.30 min	6.41×
2000	1.13 hr	35.95 min	1.88×	18.69 min	4.62×	9.76 min	6.92×
3000	3.74 hr	2.01 hr	1.86×	1.04 hr	4.60×	32.19 min	6.96×
4000	8.89 hr	4.75 hr	1.87×	2.45 hr	4.63×	1.26 hr	7.06×
5000	17.38 hr	9.30 hr	1.87×	4.77 hr	4.64×	2.44 hr	7.11×

protocols. Furthermore, it is also interesting to design *public* verifiable mechanism so that every one can verify correctness of CS's computed results via public information.

- To find an IND-CPA matrix encryption scheme with the ET-property.
- To develop secure outsourcing protocols for other core algebraic computations, such as least square computation and characteristic analysis, etc.

References

1. Mikhail J. Atallah and Keith B. Frikken. Securely outsourcing linear algebra computations. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 48–59, 2010.
2. Mikhail J. Atallah, Konstantinos N. Pantazopoulos, John R. Rice, and Eugene H. Spafford. Secure outsourcing of scientific computations. *Advances in Computers*, 54:215–272, 2001.
3. Philippe Béguin and Jean-Jacques Quisquater. Fast server-aided RSA signatures secure against active attacks. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 57–69, 1995.

4. Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In *Proceedings of the Public Key Cryptography Conference (PKC)*, pages 85–99, 2003.
5. David Benjamin and Mikhail J. Atallah. Private and cheating-free outsourcing of algebraic computations. In *Proceedings of the Annual Conference on Privacy, Security and Trust (PST)*, pages 240–245, 2008.
6. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:109, 2011.
7. Henry Carter, Benjamin Mood, Patrick Traynor, and Kevin R. B. Butler. Secure outsourced garbled circuit evaluation for mobile devices. In *Proceedings of the USENIX Security Symposium*, pages 289–304, 2013.
8. Xiaofeng Chen, Xinyi Huang, Jin Li, Jianfeng Ma, Wenjing Lou, and Duncan S. Wong. New algorithms for secure outsourcing of large-scale systems of linear equations. *IEEE Transactions on Information Forensics and Security*, 10(1):69–78, 2015.
9. Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Proceedings of EUROCRYPT*, pages 315–335, 2013.
10. Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 483–501, 2010.
11. Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 119–136, 2001.
12. Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 378–394, 2005.
13. Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative scientific computations. In *Proceedings of the IEEE Computer Security Foundations Workshop (CSFW)*, pages 273–294, 2001.
14. Wenliang Du, Yunghsiang S. Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the International Conference on Data Mining (SDM)*, pages 222–233, 2004.
15. Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 501–512, 2012.
16. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
17. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 465–482, 2010.
18. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009.
19. Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 116–137, 2010.
20. Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 264–282, 2005.
21. Seny Kamara, Payman Mohassel, and Ben Riva. Salus: a system for server-aided secure function evaluation. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 797–808, 2012.
22. Alan F. Karr, Xiaodong Lin, Ashish P. Sanil, and Jerome P. Reiter. Privacy-preserving analysis of vertically partitioned data using secure matrix products. *Journal of Official Statistics*, 25(1):125–138, 2009.
23. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 335–354, 2004.
24. Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 291–310, 2007.
25. Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In *Proceedings of the Public Key Cryptography Conference (PKC)*, pages 48–63, 2002.
26. Xinyu Lei, Xiaofeng Liao, Tingwen Huang, and Huaqing Li. Cloud computing service: the case of large matrix determinant computation. *IEEE Transactions on Services Computing*, X(X), 2014.
27. Xinyu Lei, Xiaofeng Liao, Tingwen Huang, Huaqing Li, and Chunqiang Hu. Outsourcing large matrix inversion computation to a public cloud. *IEEE Transactions on Cloud Computing*, 1(1):78–87, 2013.
28. Chae Hoon Lim and Pil Joong Lee. Security and performance of server-aided RSA computation protocols. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 70–83, 1995.
29. Payman Mohassel. Efficient and secure delegation of linear algebra. *IACR Cryptology ePrint Archive*, 2011:605, 2011.
30. Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In *Proceedings of the Advances in Cryptology (CRYPTO)*, pages 481–496, 2008.
31. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the ACM Cloud Computing Security Workshop (CCSW)*, pages 113–124, 2011.

32. Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 522–541, 2006.
33. Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
34. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of EUROCRYPT*, pages 24–43, 2010.
35. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.