# CASO: Cost-Aware Secure Outsourcing of General Computational Problems

Kai Zhou, *Student Member, IEEE* and Jian Ren, *Senior Member, IEEE*

**Abstract**—Computation outsourcing is an integral part of cloud computing. It enables end-users to outsource their computational tasks to the cloud and utilize the shared cloud resources in a pay-per-use manner. However, once the tasks are outsourced, the end-users will lose control of their data, which may result in severe security issues especially when the data is sensitive. To address this problem, secure outsourcing mechanisms have been proposed to ensure security of the end-users' outsourced data. In this paper, we investigate outsourcing of general computational problems which constitute the mathematical basics for problems emerged from various fields such as engineering and finance. To be specific, we propose affine mapping based schemes for the problem transformation and outsourcing so that the cloud is unable to learn any key information from the transformed problem. Meanwhile, the overhead for the transformation is limited to an acceptable level compared to the computational savings introduced by the outsourcing itself. Furthermore, we develop cost-aware schemes to balance the trade-offs between end-users' various security demands and computational overhead. We also propose a verification scheme to ensure that the end-users will always receive a valid solution from the cloud. Our extensive complexity and security analysis show that our proposed Cost-Aware Secure Outsourcing (CASO) scheme is both practical and effective.

**Index Terms**—Cloud computing, computation outsourcing, security, efficiency, cost-awareness

## 1 Introduction

Cloud computing paradigm provides end-users an on-demand access to a shared pool of computing resources, such as computational power and storage. It enables the end-users to utilize those resources in a pay-per-use manner instead of purchasing expensive equipment upfront. Computation outsourcing is a key component of cloud computing. It enables the resource-constrained end-users to outsource their computational tasks to the cloud servers. Then the tasks are processed in the cloud servers and solutions are returned to the end-users. The technical and economic advantages make computation outsourcing a promising application for cloud computing.

However, security has become one of the major concerns that prevent computation outsourcing from being widely adopted. When the end-users outsource their tasks to the cloud, they inevitably lose control of their own data, while the cloud servers will get full access to not only the problem itself but also the input, the intermediate computational results and the output of the problem, which may contain sensitive end-user data, such as financial statistics or health records. As a result, the end-users' privacy is totally exposed to the cloud. Furthermore, the cloud may have the motivation to cheat in the computation process thus false solutions may be returned to the end-users. This is because the computing resources are regarded as a kind of commodity and the cloud may try to reduce the cost by simply not investing enough computing resources as it has claimed. For example, the cloud may just return a trivial result for an outsourced task thus saving a lot of resources. All these issues call for designs of more secure and privacy-preserving outsourcing mechanisms that can also provide end-users the ability to validate the received results.

To address the aforementioned issues, researchers have proposed various secure outsourcing schemes for different types of computational problems, such as sequence comparison [1], [2], [3], linear algebra [4], [5], [6], [7], [8], [9], [10], modular exponentiation [11], [12], [13] and other upper-layer algorithms [14], [15], [16], [17], [18]. The techniques utilized by these schemes can be divided into two categories: encryption based schemes and disguising based schemes. Researchers from the cryptography community are trying to develop specific encryption schemes under which computation can be carried out on encrypted data. For instance, in [19] the authors proposed a fully homomorphic encryption scheme under which an arbitrary boolean circuit can be evaluated directly over the encrypted data. Based on this homomorphic encryption and Yao's garbled circuit [20], the authors in [21] designed a secure outsourcing scheme for arbitrary functions where the input and output privacy are protected and the results can be verified in a non-interactive way. However, the main drawback of this type of schemes is that they all require expensive encryption operations thus making it impractical to be carried out in the cloud scenario. Researchers in the theoretic computer science community have developed some disguising techniques to transform different types of computational problems to disguised forms so that the private information of the original

• *The authors are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824-1226. E-mail: {zhoukai, renjian}@msu.edu.*

*Manuscript received 11 Sept. 2017; revised 19 Jan. 2018; accepted 5 Mar. 2018. Date of publication 15 Mar. 2018; date of current version 7 Apr. 2021. (Corresponding author: Kai Zhou.)*
*Digital Object Identifier no. 10.1109/TSC.2018.2814991*

problems is concealed. Based on disguising, the authors in [22] and [4] developed schemes to securely outsource some basic scientific operations such as matrix multiplication, matrix inversion and convolution. More recently, secure and practical outsourcing schemes were proposed in [6], [23] for linear programming. In [24], [25], the authors focused on outsourcing of large-scale systems of linear equations. However, the above mentioned disguising techniques are specially designed for a particular kind of scientific computation, mostly lies in the scope of linear algebra. Thus the applicability of the proposed schemes is quite limited.

In this paper, we aim at developing a secure outsourcing scheme that is suitable for general computational problems. The challenges come from various aspects. First, we target at general computational problems which cover the scope of linear and non-linear problems such as system of equations (linear or non-linear), linear programming and convex optimization. Due to the different natures of these problems, it is extremely challenging to design an outsourcing scheme suitable for various kinds of computational problems. Second,the end-users are resource-constrained which means that the operations can be implemented before and after the outsourcing are quite limited. Third, the end-users vary from handheld mobile devices to desktop workstations in terms of resource constraints and security requirements. Thus it is not easy to design a scheme that can meet the requirements of various end-users. Finally, our preliminary investigation shows that a more complex pre-processing of the problem will ensure a more secure outsourcing process. However, it also creates more computational burden on the end-users. Thus there exists a trade-off between the computational complexity that the end-users can afford and the security they can get in return. All these concerns make it extremely hard to design a secure outsourcing scheme for general computational problems.

To deal with the aforementioned challenges, we propose a secure outsourcing scheme based on affine mappings. The basic idea is that before outsourcing, the independent variables of the computational problem is mapped to a new group of variables through an affine mapping. Correspondingly, the original problem is transformed to a new form that can be securely outsourced to the cloud. Then the cloud can generate valid results from the transformed problem and return the results of the transformed problem back to the end-user. By applying an inverse affine transformation on the results returned from the cloud, the end-user can derive the valid results to the original problem efficiently at the local environment.

This paper can be considered as an extension of our previous conference paper [8] that only considers outsourcing linear systems. In this paper, we extend the scope to general computational problems including non-linear systems. We also provide formal analysis of security and privacy. Especially, novel methods are proposed to characterize the privacy of outsourced data, which enables the investigation of the trade-off between security and efficiency. The contributions of this paper can be summarized as follows:

- We propose a cost-aware secure outsourcing scheme (CASO) that is generally suitable for a wide variety of computational problems, such as system of equations, linear programming and convex optimization.
- We investigate the trade-off between the computational complexity and security such that end-users can choose the most suitable outsourcing scheme according to their own resource constraints and security demands.
- Our analysis and performance comparison demonstrate that CASO is much more efficient than the existing schemes with comparable security levels.
- We also introduce a verification process which enables the end-users to verify the validity of the results returned from the cloud servers.

The rest of this paper is organized as follows. In Section 2, we introduce our system model, threat model and our design goals. In Section 3, we present the basic idea of CASO based on affine mappings. We use system of linear equations as a case study to illustrate our cost-aware design philosophies in Section 4. We extend our design to non-linear problems in Section 5 and the result verification scheme is introduced in Section 6. We evaluate the performance of our scheme by comparing it with several existing works and giving some numeric results in Section 7. We conclude our work in Section 8.

## 2 Problem Statement

### 2.1 System and Threat Model

We consider a system consisting of two entities: the end-user and the cloud. Suppose that an end-user wants to solve a general computational problem denoted by $F(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a group of independent variables. Note that $F(\mathbf{x})$ describes a general computational problem not necessarily restricted to a function. For example, it can be a system of equations or an optimization problem. However, due to lack of resources, the end-user needs to outsource the problem to the cloud which is considered to have infinite computing resources. Before outsourcing, the end-user will transform the original problem at the local side in order to prevent information leakage. On receiving the transformed problem, the cloud server will carry out the computing process and return the solution to the end-user. Then at the local side, an inverse transformation is carried out on the solution returned from the cloud to recover the solution of the original problem. Based on the transformation and the information returned by the cloud, the end-user is able to verify the validity of the received solution.

### 2.2 Design Goals

Under the above system and threat model, our proposed outsourcing scheme should achieve the following goals:

(1) *Soundness*: Given that the cloud is trustworthy, the transformation on the problem and the inverse transformation of the returned result should guarantee that the recovered solution is correct.
(2) *Security*: When the problem is outsourced to the cloud, it should be computationally infeasible for the cloud server to infer the direct information of the original outsourced problem.
(3) *Verifiability*: In case that the cloud cannot be fully trusted, the end user should have the ability to verify the validity of the solution returned by the cloud.

(4) *Efficiency*: The outsourcing scheme should be efficient in computation and communication. For computation, the overhead caused by the problem transformation, the inverse transformation and the result verification should be limited to $\mathcal{O}(n^2)$. For communication, the overhead caused by the outsourcing process should be in the same level as that of outsourcing the original problem.

(5) *Cost-Awareness*: The end-users can select different outsourcing strategies according to their own computational constraints and security demands in a cost-aware manner.

### 2.3 Application Scenarios

Our secure outsourcing scheme serves as an important building block in various high-level applications, since we focus on general computational problems serving as the underlying mathematical models in many practical problems. For example, consider a cloud-assisted image reconstruction system, where some image sensors will upload compressed image samples to the cloud. The cloud will store the image samples and help to reconstruct the image. We note that the core process in image reconstruction can be modeled as a linear programming problem [26]. To preserve the privacy of the images, the sensors can transform the image samples following the procedure in our secure outsourcing scheme. Then, the cloud will help to solve the transformed linear programming problem and returned the disguised result to the data users. At last, the data users can easily reconstruct the image based on the returned result.

## 3 Secure Outsourcing Based on Affine Mapping

### 3.1 Basic Framework

As mentioned previously, we assume that the end user has a general computational problem $F(\mathbf{x})$ to be solved. Due to the lack of resources, the end user needs to outsource $F(\mathbf{x})$ to the cloud. We formally divide the outsourcing process into the following phases.

(1) *Key Generation*: $\mathsf{KeyGen}(\lambda) \rightarrow \mathbf{S}$. In this phase, the end-user generates the secret key $\mathbf{S}$ based on the security parameter $\lambda$.

(2) *Problem Transformation*: $\mathsf{ProbTran}(\mathbf{S}, F(\mathbf{x})) \rightarrow G(\mathbf{y})$. Based on this secret key $\mathbf{S}$, the end-user transforms $F(\mathbf{x})$ to a new form $G(\mathbf{y})$, where $\mathbf{y}$ is the new input.

(3) *Cloud Computation*: $\mathsf{CloudCom}(G(\mathbf{y})) \rightarrow \{\mathbf{y}^*, \Phi\}$. On receiving the transformed problem $G(\mathbf{y})$, the cloud carries out the necessary computation and gives the solution $\mathbf{y}^*$ as well as a proof $\Phi$ of the validity of the returned solution.

(4) *Result Recovery and Verification*: $\mathsf{RecVeri}(\mathbf{y}^*, \mathbf{S}, \Phi) \rightarrow \{\mathbf{x}^*, \Lambda\}$. By utilizing the secret key $\mathbf{S}$, the end-user recovers solution $\mathbf{x}^*$ to the original problem from $\mathbf{y}^*$. Based on the proof $\Phi$, the end-user gives the decision $\Lambda = \{\mathsf{Ture}, \mathsf{False}\}$, indicating the validity of $\mathbf{x}^*$.

### 3.2 Security Characterization

In this section, we characterize the security of a secure outsourcing scheme.

First, we characterize the information of the problem to be outsourced. For a computational problem $F(\mathbf{x})$, the most sensitive information is the problem itself $F(\cdot)$ and the output $\mathbf{x}^*$. Depending on the types of the computational problem, some other information, such as the zeros and poles, could also be sensitive. In light of this, we define direct information and indirect information of a computational problem as follows.

- *Direct Information*: for a computational problem $F(\mathbf{x})$, the direct information is the problem itself $F(\cdot)$ and the output $\mathbf{x}^*$;
- *Indirect Information*: other information besides direct information is defined as indirect information.

Based on the above characterization of information, we define the security notions for an outsourcing scheme.

**Definition 1 (Security).** *An outsourcing scheme achieves security if for any given set of transformed problems $\{G(\mathbf{y}_i)\}$ and the solution $\{\mathbf{y}_i^*\}$, it is computationally infeasible for the cloud to recover the direct information.*

In the scenario of computation outsourcing, the cloud is able to observe the transformed problem, which corresponds to the ciphertext in a cryptosystem. In this sense, the cloud is able to conduct ciphertext-only attack. In the above security definition, we define security for direct information. To measure what indirect information the cloud can learn, we define the notion of privacy as follows. First, we define an experiment to model the attack by the cloud.

---

**Outsourcing Experiment** $\mathsf{Exp}_{\mathcal{A},\mathsf{out}}(\lambda)$:

- The adversary $\mathcal{A}$ outputs two computational problems $F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ of the same type.
- The challenger runs $\mathsf{KeyGen}(\lambda)$ to obtain the secret key $\mathbf{S}$.
- The challenger outputs a uniform bit $b \in 0, 1$. It runs $\mathsf{ProbTran}(\mathbf{S}, F_b(\mathbf{x}))$ to obtain the transformed problem $G_b(\mathbf{y})$.
- $\mathcal{A}$ outputs a bit $b'$.
- The output of the experiment is define as 1 if $b = b'$. Otherwise, the output is 0.

---

**Definition 2 (Privacy).** *An outsourcing scheme achieves privacy for a given security parameter $\lambda$ if for any probabilistic polynomial time adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}$ *such that*

$$|\mathsf{Pr}(\mathsf{Exp}_{\mathcal{A},\mathsf{out}}(\lambda) = 1) - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

It should be made clear that the security notions defined here is different from those for traditional cryptosystems in that the transformation does not depend on a cryptographic algorithm, even though we adopted the notions such as semantic security under ciphertext-only attack in cryptography to describe the privacy of indirect information. This is because the semantic security requires that no key information can be derived from the ciphertext, which resembles our privacy requirement that no indirect information can be learnt from the transformed problems.

**Remark 1 (Security Requirements).** The basic security is the minimum security that an outsourcing scheme should provide. That is, given the transformed problem, the cloud is unable to recover the original problem and solution (direct information). In contrast, privacy characterizes a stronger notion of security. Under the definition of privacy, the transformed problem should achieve indistinguishability. In other word, based on the transformed problem, the cloud should not be able to recover any meaningful information (indirect information).

**Remark 2 (Cost-Awareness).** The achievable privacy of an outsourcing scheme should be determined by the needs of the end-user. That is, in some scenario, an end-user may desire to achieve a strong notion of security; while in many other cases, the end-user may only need security of the direct information. On the other hand, in the practical design of secure outsourcing schemes, a stronger notion of security is achieved at the cost of a higher computational complexity. A cost-aware secure outsourcing scheme should provide an end-user the flexibility to select the most efficient outsourcing scheme that satisfies the end-user's security requirements.

### 3.3 Problem Transformation

The basic idea of problem transformation is to map the independent variables of the problem to a new group of variables such that the original problem is transformed to a new form. To be specific, suppose the original problem is $F(x)$. We assume that $\psi : \mathbb{R}^n \to \mathbb{R}^n$ is a general one-to-one mapping function. Let $\mathbf{x} = \psi(\mathbf{y})$, then $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$. In this way, the original input $\mathbf{x}$ can be transformed to input $\mathbf{y}$ with the relationship determined by the function $\psi$. Below, we give the equivalence definition of two computational problems.

**Definition 3 (Equivalence).** *Denote a set of computational problems as $\Omega = \{\Gamma \mid \Gamma : \mathbb{R}^n \to \mathbb{R}^n\}$. For any $F \in \Omega$, if there exists a one-to-one mapping $\psi : \mathbb{R}^n \to \mathbb{R}^n$ such that $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$, then $F$ is said to be equivalent to $G$. We denote it as $F \sim G$. The equivalent class of $F$ is denoted as $[F] = \{\Gamma \in \Omega \mid \Gamma \sim F\}$.*

**Theorem 1.** *The equivalence relation defined in Definition 3 is well-defined.*

**Proof.** We only need to prove that the relation defined in Definition 3 is reflexive, symmetric and transitive. First, it is obvious that for every $F \in \Omega$, if we select the one-to-one mapping $\psi$ to be the identity mapping, then we have $F(\mathbf{x}) = F(\psi(\mathbf{y})) = F(\mathbf{y})$. Thus for every $F \in \Omega$, we have $F \sim F$ which demonstrates the property of reflexivity. Second, for $F, G \in \Omega$, if $F \sim G$, then there exists a one-to-one mapping $\psi$ such that $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y}) = G(\mathbf{y})$, which indicates the existence of an inverse mapping $\psi^{-1}$ such that $G(\mathbf{y}) = (F \circ \psi)(\psi^{-1}(\mathbf{x})) = F(\mathbf{x})$. Thus we have $G \sim F$ and the property of symmetry holds. To prove the property of transitivity, assume that $F, G, H \in \Omega$ such that $F \sim G$ and $G \sim H$. This means that there are two one-to-one mappings $\psi$ and $\phi$ such that $\mathbf{x} = \psi(\mathbf{y})$, $F(\mathbf{x}) = F(\psi(\mathbf{y})) = G(\mathbf{y})$ and $\mathbf{y} = \phi(\mathbf{z})$, $G(\mathbf{y}) = G(\phi(\mathbf{z})) = H(\mathbf{z})$. Therefore, we have $F(\mathbf{x}) = F(\psi(\mathbf{y})) = F((\psi \circ \phi)(\mathbf{z})) = H(\mathbf{z})$. Since $\psi$ and $\phi$ are both one-to-one mappings, the mapping $\psi \circ \phi$ is also one-to-one. Thus from the definition we have $F \sim H$ and the equivalence relation is transitive. □

The above equivalence definition gives an insight of CASO. Based on a one-to-one mapping $\psi$, the end-user first transforms the original problem $F(\mathbf{x})$ to an equivalent form $G(\mathbf{y})$ that can be securely outsourced to the cloud. Since the solutions to the two problem satisfy $\mathbf{x}^* = \psi(\mathbf{y}^*)$, the end-user can always recover $\mathbf{x}^*$ from $\mathbf{y}^*$ returned by the cloud. Thus the essence of our proposed scheme lies in finding a proper one-to-one mapping that satisfies the various design goals.

**Definition 4.** *An affine mapping $\psi : \mathbb{R}^n \to \mathbb{R}^n$ is defined as a mapping from $\mathbf{x} \in \mathbb{R}^n$ to $\mathbf{y} \in \mathbb{R}^n$ satisfying $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is nonsingular and $\mathbf{r} \in \mathbb{R}^n$.*

It is clear that as long as $\mathbf{K}$ is nonsingular, the affine mapping defined above is a one-to-one mapping. The soundness of our proposed scheme based on affine mapping is guaranteed by the following theorem.

**Theorem 2 (Soundness).** *Under the affine mapping, the transformed problem is equivalent to the original problem. That is the end-user is guaranteed to be able to recover the valid solution of the original problem from the solution returned by the cloud.*

**Proof.** The proof of soundness follows the definition of equivalence. The affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$ is one-to-one as long as $\mathbf{K}$ is non-singular. Thus by definition, $F \sim G$ under this affine mapping. Since the solutions to the two problems satisfy $\mathbf{x}^* = \mathbf{K}\mathbf{y}^* + \mathbf{r}$, given $\mathbf{y}^*$ returned by the cloud, the end-user is able to recover $\mathbf{x}^*$ at the local side. □

**Remark 3.** Our scheme is fundamentally different from the previous schemes, such as [6] and [24]. Given a computational problem, the previous schemes try to extract the key parameters that can represent the problem, and then try to disguise these key parameters to a different form thus representing a different computational problem so that the original problem is protected from the cloud. While it is relatively easy to extract and disguise the key parameters of a linear computational problem (e.g., linear programming and system of linear equations), it is hard for non-linear problems, which limits the previous schemes to only linear problems such as linear programming and systems of linear equations.

In comparison, our scheme starts from the variables since in essence, a computational problem is about computation of the variables. We map the group of the original variables to another group of variables in such a way that the secret information is protected. When we map the variables $\mathbf{x}$ to a new group of variables $\mathbf{y}$ through $\mathbf{x} = \psi(\mathbf{y})$, the original problem becomes $F(\mathbf{x}) = F(\psi(\mathbf{y})) = (F \circ \psi)(\mathbf{y})$, which can naturally be applied to both linear and non-linear problems.

## 4 Cost-Aware Design for Linear Systems

In this section, we present our cost-aware secure outsourcing scheme for general computational problems. In the region of linear computation, we deploy system of linear equations as a case study to show the principles of our design. Then we show that the proposed CASO can be well extended to linear programming.

TABLE 1
Summary of Key Notations

| Symbol | Interpretation |
|---|---|
| $F(\mathbf{x})$ | original problem on variables $\mathbf{x}$ |
| $G(\mathbf{y})$ | transformed problem on variables $\mathbf{y}$ |
| $n$ | number of independent variables |
| $\mathbf{K}, \mathbf{r}$ | one-time transformation key |
| $\mathbf{A}$ | coefficient matrix |
| $\mathbf{A}'$ | transformed coefficient matrix |
| $W$ | bandwidth of a band matrix |
| $\theta$ | upper bound of non-zeros in each row or column of a sparse matrix |
| $N$ | number of terms in a non-linear system |
| $L$ | number of polynomials in a non-linear system |
| $\mathcal{T}_e$ | user-side computational time with outsourcing |
| $\mathcal{T}_s$ | user-side computational time without outsourcing |
| $\mathcal{I}$ | computational gain from outsourcing |

### 4.1 Outsourcing Scheme

In the problem transformation phase, the end-user first generates a *random one-time secret key* $\mathbf{S} = \{\mathbf{K}, \mathbf{r}\}$, where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is a non-singular matrix and $\mathbf{r} \in \mathbb{R}^n$. Then $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$ is a one-to-one mapping from $\mathbf{x}$ to $\mathbf{y}$. The key $S$ will be discarded after each use. The randomness of the key selection ensures that it is very unlikely for any key to be reused.

Suppose the computational problem is a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ and $\mathbf{A}$ is an $n \times n$ non-singular matrix. The function $\mathsf{ProbTran}(\mathbf{S}, F(\mathbf{x})) \rightarrow G(\mathbf{y})$ takes the secret key $\mathbf{S} = \{\mathbf{K}, \mathbf{r}\}$ and the linear system as input and generates the output as $\mathbf{A}\mathbf{K}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{r}$. Denote $\mathbf{A}' = \mathbf{A}\mathbf{K}$ and $\mathbf{b}' = \mathbf{b} - \mathbf{A}\mathbf{r}$ and the system is transformed to $G(\mathbf{y}) : \mathbf{A}'\mathbf{y} = \mathbf{b}'$ which can be outsourced to the cloud.

In the phase of cloud computation, the cloud solves $G(\mathbf{y})$ utilizing the typical methods and returns the solution $\mathbf{y}^*$ to the end-user. Then in the result recovery phase, the end-user recovers the solution to the original system of linear equations as $\mathbf{x}^* = \mathbf{K}\mathbf{y}^* + \mathbf{r}$. In the following sections, we will discuss the detailed design of our secure outsourcing scheme. Some of the key notations are listed in Table 1.

### 4.2 Design Analysis

From the above outsourcing scheme, we can see that the computational overhead for the end-user incurs both in the problem transformation and the result recovery phase. To be more specific, in the problem transformation phase, the end-user needs to calculate $\mathbf{A}\mathbf{K}$ and $\mathbf{A}\mathbf{r}$. To recover the original solution $\mathbf{x}^*$ from the received solution $\mathbf{y}^*$, the end-user has to calculate $\mathbf{K}\mathbf{y}^*$. Among those operations, the matrix multiplication $\mathbf{A}\mathbf{K}$ is the most computationally expensive one. Thus in our discussion, we will analyze the number of multiplications $M$ required to compute $\mathbf{A}\mathbf{K}$. In the following analysis, we denote $\mathbf{A} = \{a_{ij} | i, j = 1, 2, \ldots, n\}$ and $\mathbf{K} = \{k_{ij} | i, j = 1, 2, \ldots, n\}$.

To multiply two arbitrary $n \times n$ matrices, the typical complexity is $\mathcal{O}(n^3)$, which is generally believed to be too high and unacceptable for mobile client computation. However, in our design, we can actually control the complexity by selecting matrix $\mathbf{K}$ properly so that the computational complexity can be effectively reduced without compromising security. Since matrix multiplication is the most expensive part of the end-user's processing, our goal is to ensure that the complexity of multiplying $\mathbf{K}$ with an arbitrary matrix $\mathbf{A}$ is bounded by $\mathcal{O}(n^2)$, which is within the end-user's computational constraints.

In the following sections, we provide four schemes with different types of non-singular secret key $\mathbf{K}$ based on the above described complexity constraints.

#### 4.2.1 $\mathbf{K}$ *is a Diagonal Matrix (Scheme-1)*

A diagonal matrix $\mathbf{K}$ has the format $\mathbf{K} = \{k_{ij} | k_{ij} = 0, \forall i \neq j\}$. Since $\mathbf{K}$ must be non-singular, all the entries in the diagonal have to be non-zero numbers. When $\mathbf{K}$ is a diagonal matrix, we have $M = n^2$.

#### 4.2.2 $\mathbf{K}$ *is a Permutation Matrix (Scheme-2)*

A permutation matrix $\mathbf{K}$ has exactly one non-zero entry in each row and each column in the matrix. When $\mathbf{K}$ is a permutation matrix, we have $M = n^2$.

#### 4.2.3 $\mathbf{K}$ *is a Band Matrix (Scheme-3)*

Suppose the band matrix $\mathbf{K}$ has an upper half-bandwidth $p$ and a lower half-bandwidth $q$ such that $k_{ij} = 0$ for $i > j + p$ and $j > i + q$. The total bandwidth of $\mathbf{K}$ is denoted by $W = p + q + 1$. When $\mathbf{K}$ is a band matrix, for simplicity, we assume that $\mathbf{K}$ has an equal upper and lower half-bandwidth $p = q = \omega$, then $W = 2\omega + 1$, and the number of multiplications $M$ can be calculated as $M = (2\omega + 1)n^2 - (\omega^2 + \omega)n$.

#### 4.2.4 $\mathbf{K}$ *is a Sparse Matrix (Scheme-4)*

Suppose $\mathbf{K}$ is a sparse matrix. The density $d$ is defined as the ratio of non-zero elements in the matrix. We assume that the number of non-zero elements in each row and each column of $\mathbf{K}$ is up-bounded by a constant $\theta$. When $\mathbf{K}$ is a sparse matrix, it is usually stored in a special manner such as Dictionary of Keys (DOK) [27] in computation. Thus the complexity of matrix multiplication can be approximately measured by the number of non-zero elements, which is $dn^3$ in our discussion. Since we have assumed that $d \leq \frac{\theta}{n}$, the number of multiplication becomes $M = \theta n^2$.

In summary, through the above analysis, we demonstrate that for the four proposed schemes, the complexity of multiplying $\mathbf{K}$ with an arbitrary matrix $\mathbf{A}$ is $\mathcal{O}(n^2)$. Since matrix multiplication is the most expensive part of the end-user's processing, we can derive that the overall computational complexity for the end-user is $\mathcal{O}(n^2)$, which is within the end-user's computational constraints.

### 4.3 Security Analysis

In this section, we will analyze the security of our proposed CASO. We will focus on the security of the coefficient matrix $\mathbf{A}$ of the original function $F(\mathbf{x})$, the variable $\mathbf{x}$ in the function $F(\mathbf{x})$ and the form of the function $F(\mathbf{x})$.

**Theorem 3.** *CASO can ensure security of the direct information. In other words, for the four schemes in CASO, it is computationally infeasible for the cloud to recover the original coefficient matrix $A$ and the output $\mathbf{x}^*$ for the system of linear equations.*

**Proof.** For a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, the original problem is represented by the matrix $\mathbf{A}$ and the vector $\mathbf{b}$. The output is $\mathbf{x}^*$, which is the solution of the system. Under the affine mapping, the system of equations is

transformed to $\mathbf{A}'\mathbf{y} = \mathbf{b}'$, where $\mathbf{A}' = \mathbf{AK}$ and $\mathbf{b}' = \mathbf{b} - \mathbf{Ar}$. Therefore, it is computationally infeasible for the cloud to recover $\mathbf{A}$ and $\mathbf{b}$ from $\mathbf{A}'$ and $\mathbf{b}'$ since both $\mathbf{K}$ and $\mathbf{r}$ are only used once and kept secret at the local side. Additionally, since the original solution is recovered by $\mathbf{x}^* = \mathbf{Ky}^* + \mathbf{r}$, without knowing $\mathbf{K}$ and $\mathbf{r}$, the cloud cannot recover $\mathbf{x}^*$. In this way, the output of the system is concealed. Thus, all the four schemes are secure in outsourcing the system of linear equations. □

**Theorem 4.** *CASO can achieve the privacy of output* $\mathbf{x}^*$.

**Proof.** From the definition of privacy, an end-user plays the role of the challenger and generates the secret key $(\mathbf{K}, \mathbf{r})$. An adversary $\mathcal{A}$ submits two outputs $\mathbf{y}_0$ and $\mathbf{y}_1$ to the end-user, The end-user generates a random bit $b$ and transforms $\mathbf{y}_b$ to $\mathbf{x}_b = \mathbf{Ky}_b + \mathbf{r}$ and sends $\mathbf{x}_b$ back to the adversary. The task of the adversary is to output another bit $b'$. If $|\mathsf{Prob}(b = b') - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, the adversary $\mathcal{A}$ will loose the game and it is proved that CASO can achieve the privacy of output $\mathbf{x}^*$. Note that $\mathbf{r}$ is randomly generated. As a result, regardless of the selection of $\mathbf{K}$, $\mathbf{x}_b = \mathbf{Ky}_b + \mathbf{r}$ is random. Thus, the advantage for the adversary to distinguish $\mathbf{x}_0$ and $\mathbf{x}_1$ is negligible. In other words, the adversary can only generate a bit $b'$ such that $|\mathsf{Prob}(b = b') - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$. □

It is worth to mention that all the four schemes in CASO can successfully conceal the zeros and poles of the function since zeros and poles are information of the variables $\mathbf{x}$.

**Remark 4.** The complete privacy of the coefficient matrix $\mathbf{A}$ is unachievable under affine mapping. This is because the adversary can always distinguish $\mathbf{A}'_0 = \mathbf{A}_0\mathbf{K}$ from $\mathbf{A}'_1 = \mathbf{A}_1\mathbf{K}$. For example, the adversary can select $\mathbf{A}_0$ and $\mathbf{A}_1$ such that one of these two matrix is singular. Then the rank of the retuning matrix $\mathbf{A}'_b$ would be different.

To this end, we have shown that the four schemes in CASO is able to achieve security and the privacy of the output $\mathbf{x}^*$. However, the privacy information of the coefficient matrix $\mathbf{A}$ s not fully achievable. In the following analysis, we will show different protection of indirect information provided by the four schemes in CASO.

**Theorem 5.** *Suppose $\psi$ is a rational mapping, meaning that $\psi$ can be represented as a quotient of two polynomial functions, $G = F \circ \psi$, then we have the following results:*

*(1) If $F$ is a rational function, then $G$ is rational.*
*(2) If $F$ is an irrational function, then $G$ is irrational.*

**Proof.** Since $\psi$ is a rational mapping, we assume $\psi(x) = \frac{P(x)}{Q(x)}$, where $P(x)$ and $Q(x)$ are polynomials. When $F$ is a rational function, suppose

$$F(x) = \frac{f_1(x)}{f_2(x)}, \tag{1}$$

where $f_1(x) = a_0 + a_1x + \cdots + a_nx^n$, and $f_2(x) = b_0 + b_1x + \cdots + b_mx^m$. Then

$$(F \circ \psi)(x) = \frac{f_1(\psi(x))}{f_2(\psi(x))}. \tag{2}$$

Without loss of generality, we assume that $m > n$. Then we have

$$(F \circ \psi)(x) = \frac{Q^m(x) \cdot f_1(\psi(x))}{Q^m(x) \cdot f_2(\psi(x))}. \tag{3}$$

It is clear that both $Q^m(x) \cdot f_1(\psi(x))$ and $Q^m(x) \cdot f_2(\psi(x))$ are polynomials. Therefore, $F \circ \psi$ is the quotient of two polynomials and the composition $G = F \circ \psi$ is a rational function.

When $F$ is irrational, the composition $G = F \circ \psi$ cannot be rational. Otherwise, there exists an inverse rational function $\psi^{-1}$ such that $F = G \circ \psi^{-1} = F \circ \psi \circ \psi^{-1}$ becomes rational. Hence, $G = F \circ \psi$ is irrational when $F$ is irrational. □

Since the proposed affine mapping is rational, we have the following corollary.

**Corollary 1.** *Under an affine mapping $\psi$, the rationality of the function $G$ is the same as the original function $F$.*

Theorem 5 and Corollary 1 state that the rationality of the function $F$ cannot be changed through composition with a rational mapping or an affine mapping $\psi$. That is, if the function $F$ is rational, after the composition $G = F \circ \psi$, the transformed function $G$ is still rational. If $F$ is irrational, $G$ is still irrational. As a result, the side information that is related to the specific form of the function $F$ (e.g., $\sin(\cdot)$ or $\log(\cdot)$ ) may not be fully concealed by an affine mapping or even a rational mapping.

Now, we will analyze the indirect information that can be revealed by the coefficient matrix $\mathbf{A}$ of the four schemes. Under an affine mapping, the coefficient matrix $\mathbf{A}$ is transformed to $\mathbf{A}' = \mathbf{AK}$. Thus the problem is to characterize the indirect information of $\mathbf{A}$ given $\mathbf{A}'$. Let $a_{ij}, a'_{ij}$ and $k_{ij}$ be the entries of $\mathbf{A}$, $\mathbf{A}'$ and $\mathbf{K}$, respectively. By affine mapping, the entries $a'_{ij}$ are actually linear combinations of $a_{ij}$ and $k_{ij}$. In our settings, we elaborately select some of the entries in $\mathbf{K}$ to be zeros to reduce the computational complexity. Thus, in a high level view, multiplying $\mathbf{A}$ with $\mathbf{K}$ results in the combined effect of scaling and permuting of the columns of $\mathbf{A}$. In light of this, to characterize the effect of scaling and permuting, we introduce the *ratio privacy* concerning the ratio information of the entries of $\mathbf{A}$ and the *position privacy* concerning the composition of each entry of $\mathbf{A}'$ from entries of $\mathbf{A}$. In the following, we will analyze to what extent the four schemes can achieve the privacy. For scheme-$i$ $(i = 1, 2, 3, 4)$, we denote the secret key utilized in the scheme as $\mathbf{K}_i$.

For scheme-1, the secret key $\mathbf{K}$ is a diagonal matrix denoted by $\mathbf{K}_1 = \{k_{ij} | k_{ij} = 0, \forall i \neq j\}$. The entry $a'_{ij}$ in $\mathbf{A}'$ can be calculated as $a'_{ij} = k_{ii}a_{ij}$. By investigating $\mathbf{A}'$, it is obvious that each column in $\mathbf{A}'$ is related in a simple way to that in $\mathbf{A}$ such that the $i$th column in $\mathbf{A}'$ is the multiplication of the $i$th column in $\mathbf{A}$ with $k_{ii}$. In this way, only based on $\mathbf{A}'$, the cloud can easily know the ratio between any two entries within the same column in $\mathbf{A}$. Moreover, it is also clear how each entry in $\mathbf{A}'$ is composed.

For $\mathbf{K}_2$ to be a permutation matrix in scheme-2, the difference is that $\mathbf{A}'$ in scheme-2 can be regarded as the result of permuting the columns of $\mathbf{A}'$ obtained from scheme-1. Thus, although the cloud can get a knowledge of the ratio between two entries in the same column of $\mathbf{A}$, it is not sure

TABLE 2
Complexity and Security of Each Scheme : ✓ Denotes Security can be Guaranteed or Privacy can be Preserved

| Scheme | Complexity | Security | Privacy of $\mathbf{x}$ | Ratio Privacy of $\mathbf{A}$ | Position Privacy of $\mathbf{A}$ |
|---|---|---|---|---|---|
| Diagonal matrix $\mathbf{K}_1$ | $n^2$ | ✓ | ✓ | × | × |
| Permutation matrix $\mathbf{K}_2$ | $n^2$ | ✓ | ✓ | × | ✓ |
| Band matrix $\mathbf{K}_3$ | $Wn^2$ | ✓ | ✓ | ✓ | × |
| Sparse matrix $\mathbf{K}_4$ | $\theta n^2$ | ✓ | ✓ | ✓ | ✓ |

× *denotes privacy cannot be preserved.*

which particular column those two entries belong to. As a result, while scheme-2 can achieve position privacy, it may leak ratio privacy.

In scheme-3, for $\mathbf{K}_3$ to be a band matrix with upper half-bandwidth and lower half-bandwidth both equal $\omega$, it can be calculated that

$$a'_{ij} = \sum_{r=j-\omega}^{j+\omega} a_{ir} k_{rj}. \tag{4}$$

Since each entry in $\mathbf{A}'$ is a linear combination of entries in $\mathbf{A}$ and $\mathbf{K}$, the ratio information of entries in $\mathbf{A}$ is concealed. However, the disadvantage is that the cloud can still learn how a particular entry in $\mathbf{A}'$ is composed. For example, suppose $\omega = 1$, the cloud can know for sure that $a'_{ij} = a_{i(j-1)}k_{(j-1)j} + a_{ij}k_{jj} + a_{i(j+1)}k_{(j+1)j}$. In this sense, while scheme-3 can achieve ratio privacy, it may leak position privacy.

At last, for $\mathbf{K}_4$ to be a sparse matrix in scheme-4, we assume that there are exactly $\theta$ non-zero entries in each row and column of $\mathbf{K}$. Similar to scheme-3, the ratio information of entries in $\mathbf{A}$ can be concealed. Moreover, since the non-zero entries are randomly positioned in the sparse matrix $\mathbf{K}$, the cloud is unable to know how each entry in $\mathbf{A}'$ is composed. Thus, scheme-4 can achieve both ratio privacy and position privacy.

In summary, we categorized the privacy of the coefficient matrix $\mathbf{A}$ into ratio privacy and position privacy. Such categorization stems from the essence of matrix multiplication. In a high level view, multiplying $\mathbf{A}$ with a specially designed secret matrix $\mathbf{K}$ can be separated into two critical operations: weighted sum of the entries of $\mathbf{A}$ and random permutation. The former operation preserves the ratio privacy while the latter operation preserves the position privacy. Moreover, the number of non-zero entries in $\mathbf{K}$ determines to what degree the ratio privacy is preserved. However, as long as the positions of the non-zeros entries are random, the position privacy of $\mathbf{A}$ are preserved. We summarize the computational complexity and security of CASO in Table 2.

### 4.4 Trade-Off Between Complexity and Security

From the above complexity and security analysis, we can see that there is a trade-off between the computational complexity and security. As the simple scheme, scheme-1 is able to protect the original coefficient matrix while exposing the ratio between any two entries in the same column. In comparison, scheme-2 is slightly more expensive (e.g., the positions of the non-zero entries have to be stored), but it is this cost for non-zero entries' random positions that makes it effective to conceal the ratio information. The complexity of scheme-3 and scheme-4 is linearly dependent on $W$ and $\theta$, respectively. They are more costly than scheme-1 and scheme-2. However, the transformed matrix $\mathbf{A}'$ can conceal $\mathbf{A}$ and $\mathbf{K}$ in a more complex way since it can conceal the structure of the coefficient matrix. In summary, from scheme-1 to scheme-4, the security levels that they can provide increase at a cost of computational power.

In the context of cloud computing, the end-users vary from mobile devices to powerful workstations thus having different computational constraints as well as different security demands. Thus CASO provides end-users with the flexibility to choose the outsourcing schemes that are most suitable for them. These four schemes give cost-aware outsourcing for end-users to address the various security demands and computational constraints.

### 4.5 Application to Linear Programming

In this section, we will demonstrate that our design and analysis for system of linear equations can be well applied to many computational problems, such as linear programming. We consider a linear programming problem denoted by

$$F(\mathbf{x}) := \begin{cases} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{D}\mathbf{x} \geq \mathbf{0}, \end{cases} \tag{5}$$

where $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{D} \in \mathbb{R}^{s \times n}$ $(m, s \leq n)$.

Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the problem is transformed to

$$G(\mathbf{y}) := \begin{cases} \text{minimize} & \mathbf{c}^T\mathbf{K}\mathbf{y} + \mathbf{c}^T\mathbf{r} \\ \text{subject to} & \mathbf{A}\mathbf{K}\mathbf{y} = \mathbf{b} - \mathbf{A}\mathbf{r} \\ & \mathbf{D}\mathbf{K}\mathbf{y} \geq -\mathbf{D}\mathbf{r}, \end{cases} \tag{6}$$

from which we can see that the original coefficient matrix can be concealed by the secret key $\mathbf{K}$ and $\mathbf{r}$. It is obvious that the computational bottleneck lies in the multiplication of $\mathbf{K}$ with $\mathbf{A}$ and $\mathbf{D}$. Thus the same complexity and security analysis for systems of linear equations applies for linear programming. That is the complexity of the previous four schemes is all bounded by $\mathcal{O}(n^2)$. In terms of security, the four schemes are all secure in protecting the original coefficient matrix while providing different levels of protection of the side information.

In the next section, we explore the differences for non-linear computation by investigating system of non-linear equations and convex optimization problems.

## 5 Extension to Non-Linear Systems

In this section, we aim at exploring the different design issues between linear and non-linear computation. We consider a system of non-linear equations denoted by $F(\mathbf{x}) = \mathbf{0}$, where $F(\mathbf{x}) = \{f_i(\mathbf{x}) | f_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, n\}$. Typically, it is

hard to obtain a symbolic solution for the system. Thus the normal method is to solve the system of equations numerically in an iterative way. The main idea is that given a solution $\mathbf{x}_k$ in the $k$th iteration, we need to solve the linear system $\partial F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}(\mathbf{x}_{k+1}-\mathbf{x}_k) = -F(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}$, where $\partial F(\mathbf{x})$ is the Jacob matrix of $F(\mathbf{x})$. Then we can obtain the solution $\mathbf{x}_{k+1}$ in the $(k+1)$th iteration. The iteration will terminate when $\|F(\mathbf{x}^*)\| < \varepsilon$, where $\varepsilon$ is the error tolerance and $\mathbf{x}^*$ is the final solution. To minimize the communication overhead and the energy consumption of the end-users, our goal is to design off-line scheme so that the end-users are not required to interact with the cloud except the problem outsourcing and result retrieving process. In this way, the end-users only need to focus on the high level view of the problem without knowing the details of problem solving process. The detailed design and analysis of the outsourcing scheme are presented as follows.

### 5.1 Outsourcing Scheme

Compared with outsourcing of the system of linear equations, the main difference lies in the problem transformation phase. First, to start the iteration at the cloud side, an initial guess of the solution should also be outsourced. We assume that at the local side, the end-user generates an initial solution $\mathbf{x}_0$. Then with the affine mapping, the outsourced initial solution becomes $\mathbf{y}_0 = \mathbf{K}^{-1}(\mathbf{x}_0 - \mathbf{r})$. We should notice that there is an inversion operation on $\mathbf{K}$ which will impose more constraints on our selection of $\mathbf{K}$ in terms of computational complexity. Second, after substituting $\mathbf{x}$ with $\mathbf{y}$, the problem should be further transformed. We use a simple example to illustrate this point. Suppose we want to solve a system of non-linear equations

$$F(\mathbf{x}) := \begin{cases} \sin(3x_1) + 4x_2^2 + x_2x_3 = 0 \\ 2x_1 + e^{3x_2} + 2x_3^3 = 0 \\ \lg(5x_1) + \frac{1}{2x_2+1} + 3(x_3+1)^2 = 0. \end{cases} \tag{7}$$

We take the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, where $\mathbf{r} = \mathbf{0}$ and

$$\mathbf{K} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}.$$

Then the system is transformed to

$$G(\mathbf{y}) := \begin{cases} \sin(9y_1) + 16y_2^2 + 8y_2y_3 = 0 \\ 6y_1 + e^{6y_2} + 128y_3^3 = 0 \\ \lg(15y_1) + \frac{1}{4y_2+1} + 48y_3^2 + 24y_3 = -3. \end{cases} \tag{8}$$

It is obvious that to protect the cloud from revealing information from the transformed system, it is sufficient to mix the coefficient of each term in the equations with the key entry. To be specific, we assume that there are $\pi_i$ terms in equation $f_i(\mathbf{x})$ and each term is denoted by $f_i^j(\mathbf{tx})$, where $\mathbf{t}$ is the coefficient. Then each equation in the system can be written as

$$f_i(\mathbf{x}) = \sum_{j=1}^{\pi_i} f_i^j(\mathbf{tx}).$$

Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, $f_i^j(\mathbf{tx})$ is transformed to

$$g_i(\mathbf{y}) = f_i(\mathbf{K}\mathbf{y} + \mathbf{r}) = \sum_{j=1}^{\pi_i} f_i^j(\mathbf{t}(\mathbf{K}\mathbf{y} + \mathbf{r})).$$

Thus the coefficient $\mathbf{t}$ is concealed by $\mathbf{K}$ and $\mathbf{r}$, which is similar to the case of the system of linear equations. However, as illustrated in the example, the multiplication cannot be simply carried out when $f_i^j(\cdot)$ is a polynomial. Thus a further transformation is needed to mix $\mathbf{t}$ with $\mathbf{K}$ and $\mathbf{r}$ for polynomials.

Without loss of generality, we assume that the polynomial is denoted by $t_i x_i^m$ and in the affine mapping, $\mathbf{K}$ is a band matrix with bandwidth $W = 3$ and $\mathbf{r} = \mathbf{0}$. Thus under the affine mapping, the polynomial is transformed to

$$t_i(k_{i-1}y_{i-1} + k_iy_i + k_{i+1}y_{i+1})^m.$$

To mix the coefficient $t_i$ with the secret keys, one straightforward way is to expand the polynomial and then multiple it with $t_i$. However, the complexity is unacceptable for high order polynomials. Instead, we propose that it is sufficient to split the secret keys as $k_s = pq_s$, where $s = i-1, i, i+1$ such that

$$\begin{aligned} &t_i(k_{i-1}y_{i-1} + k_iy_i + k_{i+1}y_{i+1})^m \\ &\quad = t_i(pq_{i-1}y_{i-1} + pq_iy_i + pq_{i+1}y_{i+1})^m \\ &\quad = t_ip^m(q_{i-1}y_{i-1} + q_iy_i + q_{i+1}y_{i+1})^m. \end{aligned}$$

In this way, the coefficient $t_i$ in the original function and the secret keys $k_i$ are concealed.

### 5.2 Complexity Analysis

From the analysis above, we can see that the complexity of the problem transformation mainly depends on two aspects. One is the specific form of the equations, that is the number of polynomials in the equations. The other one is how $\mathbf{x}$ and $\mathbf{y}$ are related, which is determined by the number of non-zero entries in $\mathbf{K}$.

For a given system of non-linear equations, suppose that there are $N$ terms in total in the systems, among which $L$ are polynomials with orders no greater than $m$. Assume that the number of non-zero entries in $\mathbf{K}$ is up-bounded by $\lambda$ (i. e. each $x$ is substituted by at most $\lambda$ $y$'s). Thus for each non-polynomial term, the transformation takes $\lambda$ multiplications between the coefficient of the term and the key entries. And for a polynomial term $t_i x_i^m$, we assume that it is replaced by

$$\begin{aligned} t_i(k_1y_1 + \cdots + k_\lambda y_\lambda)^m &= t_i(pq_1y_1 + \cdots + pq_\lambda y_\lambda)^m \\ &= t_ip^m(q_1y_1 + \cdots + q_\lambda y_\lambda)^m. \end{aligned}$$

Then the operations involved in the transformation include one multiplication, $\lambda$ division and raising $p$ to the power of $m$. As stated previously, we utilize the number of multiplication as a measurement for complexity. We assume that in terms of computational complexity, one division is equal to one multiplication and with the method of exponentiation by squaring, the computation for $m$th power takes $\log^2 m$ multiplications. Thus, for a system of non-liner equations with $N$ terms among which $L$ are polynomials,

TABLE 3
Complexity for System of Non-Linear Equations

| Scheme | Complexity |
|---|---|
| Diagonal matrix | $N + (\log^2 m + 1)L$ |
| Permutation matrix | $N + (\log^2 m + 1)L$ |
| Band matrix | $WN + (\log^2 m + 1)L$ |
| Sparse matrix | $\theta N + (\log^2 m + 1)L$ |

the complexity can be calculated as

$$\lambda N + (\log^2 m + 1)L.$$

It is obvious that the complexity depends on $\lambda$ which is further determined by the selection of $\mathbf{K}$. We summarize the complexity of the four different types of matrices in Table 3. We can see from the table that the complexities of all schemes are constrained to $\mathcal{O}(N)$, where $N$ is the number of terms in the system of non-linear equations. Notice that typically for a system of equations, the number of terms $N$ is in the level of $n^2$, where $n$ is the number of independent variables. Thus the complexity is still bounded by $\mathcal{O}(n^2)$, which fulfills our design goals.

### 5.3 Security Analysis

Similar to the security analysis for linear systems, all of the proposed four schemes are secure in protecting the coefficient matrix, the zeros, poles and optimums of the outsourced problem. As stated in Corollary 1, CASO cannot conceal the specific form of the functions. For instance, in the example given in Section 5.1, the original system of equations is transformed to $G(\mathbf{y})$ such that the coefficients in each term of the function are changed. However, the specific forms of the function (e.g., $\sin(\cdot), \lg(\cdot)$, etc. ) remain unchanged.

For the four schemes, generally as the complexity increases, more side information can be concealed from the cloud. Different from the linear equations, a non-linear function $f_i(x)$ may contain some side information, such as maximum or minimum value which is important in some applications. For instance, the plot of the function or the extreme values may expose the distribution of the incidence of a disease among different age groups. For scheme-1 and scheme-2, the curve of the function is just a scaled version. Though scheme-2 provides better protection since it can conceal the independent variables. In scheme-3 and scheme-4, each independent variables in the original problem is substituted by several new variables. Thus the side information, such as the curve and the extreme values can be perfectly concealed.

### 5.4 Application to Convex Optimization

In this section, we show that the above schemes and analysis can also be applied to convex optimization. Convex optimization is widely employed in various practical problems. We consider a convex optimization problem denoted by

$$F(\mathbf{x}) := \begin{cases} \text{minimize} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \leq 0, i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, j = 1, \ldots, t, \end{cases} \quad (9)$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 0, \ldots, m$ and $h_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, t$ are all convex functions. Under the affine mapping $\mathbf{x} = \mathbf{K}\mathbf{y} + \mathbf{r}$, the original problem $F(\mathbf{x})$ is transformed to

$$G(\mathbf{y}) := \begin{cases} \text{minimize} & f_0(\mathbf{K}\mathbf{y} + \mathbf{r}) \\ \text{subject to} & f_i(\mathbf{K}\mathbf{y} + \mathbf{r}) \leq 0, i = 1, \ldots, m \\ & h_j(\mathbf{K}\mathbf{y} + \mathbf{r}) = 0, j = 1, \ldots, t. \end{cases} \quad (10)$$

Since the key matrix $\mathbf{K}$ and $\mathbf{r}$ are randomly generated and kept secret at the local side, the coefficient matrix of the outsourced problem is perfectly protected. And because the functions $f_i(\cdot)$ and $h_j(\cdot)$ are all non-linear functions, the security and the complexity analysis of system of non-linear equations can be well applied in this case. Thus we conclude that our outsourcing scheme is also applicable to convex optimization problems.

## 6 Results Verification

The general idea of our proposed verification scheme is to transform the problem with two independent affine mappings and outsource the two transformed problems to the cloud. Then the end-user is able to verify whether the two results returned by the cloud match with each other. We note that such a verification scheme is different from those that requires two rounds of communications. In our scheme, the end-user does not need to wait for the result of the first round outsourcing before sending out the second transformed problem. This is because the results for the two transformed problems are received simultaneously. To be specific, under the affine mappings $\mathbf{x} = \mathbf{K}_1\mathbf{y} + \mathbf{r}_1$ and $\mathbf{x} = \mathbf{K}_2\mathbf{z} + \mathbf{r}_2$, the original problem $F(\mathbf{x})$ is transformed to $G(\mathbf{y})$ and $H(\mathbf{z})$ which are outsourced to the cloud. Then the cloud solves the two outsourced problems and returns the corresponding results $\mathbf{y}^*$ and $\mathbf{z}^*$. Since the condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds for these two results, the end-users can utilize it as a criterion to verify whether the returned results are valid.

### 6.1 System of Non-Linear Equations

The idea introduced above can be applied to system of equations directly. When $F(\mathbf{x})$ is a system of linear equations, it is sufficient to verify directly whether $\|\mathbf{A}\mathbf{x}^*\| < \varepsilon$, where $\|\cdot\|$ denotes the euclidean norm of a vector and $\varepsilon$ is a pre-defined error tolerance. The complexity of this verification process is $\mathcal{O}(n^2)$.

When $F(\mathbf{x})$ is a system of non-linear equations, since the end-user will have to evaluate the non-linear functions, the computational cost for direct verification generally exceeds $\mathcal{O}(n^2)$. However, based on our idea of outsourcing twice, the end-user only needs to check the condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$. Since the verification process involves only linear operations, the computational complexity is bounded by $\mathcal{O}(n^2)$. As system of equations is typically solved by iterative method, the solution is not accurate. Thus we may need to change the equality condition to

$$\|(\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1) - (\mathbf{K}\mathbf{z}^* + \mathbf{r}_2)\| < \varepsilon.$$

In the following analysis, we uniformly utilize the equality condition $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ as the verification criteria. When the computational problems are solved inaccurately,

the equality condition should be changed to its inequality variation.

### 6.2 Optimization Problems

When $F(\mathbf{x})$ is an optimization problem, we utilize convex optimization as an example to illustrate the verification process. And it can be easily applied to other optimization problems, such as linear programming. The output of a convex optimization problem can be divided into three cases: normal, infeasible and unbounded [28, Chapter 4.1]. For the convex optimization problem defined in Equation (9), the *domain* $\mathcal{D}$ is the set for which the objective function and the constraint functions are defined. That is

$$\mathcal{D} = \bigcap_{i=1}^{m} \mathbf{dom} f_i \cap \bigcap_{i=i}^{t} \mathbf{dom} h_i.$$

The *feasible set* is $\mathcal{E} = \{\mathbf{x} \in \mathcal{D} \mid f_i(\mathbf{x}) \leq 0, i = 1, \ldots, m, h_i(\mathbf{x}) = 0, i = 1, \ldots, t\}$. In the normal case, there exists an optimal point $\mathbf{x}^* \in \mathcal{E}$ such that $f_0(\mathbf{x}^*) \leq f_0(\mathbf{x}), \forall \mathbf{x} \in \mathcal{E}$. In the infeasible case, $\mathcal{E} = \emptyset$. In the unbounded case, there exists points $\mathbf{x}_k \in \mathcal{E}$ such that $f_0(\mathbf{x}_k) \to -\infty$ as $k \to \infty$.

For the cloud to cheat, it must return results in the same case for the two outsourced problem $G(\mathbf{y})$ and $H(\mathbf{z})$ as mentioned above. Suppose that $\mathbf{y}^*$ and $\mathbf{z}^*$ are the two returned results and they belong to the same case. In the following, we will present the verification scheme for the three different cases separately.

#### 6.2.1 Normal Case

The above proposed verification scheme works well for the normal case. That is if the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds, the end-user can make sure that a valid result can be recovered. This is because whatever the correct result is (normal, infeasible or unbounded), the cloud is not able to come up with two results that satisfy the equality without actually conducting the computation process. And this verification process for normal case forms the basis for the verification for other cases.

#### 6.2.2 Infeasible Case

The above verification scheme would fail if the cloud simply returns an infeasible result for any outsourced convex optimization problem. To deal with this issue, we utilize phase I method as described in [28, Chapter 11] to check the feasibility of the problem. For a convex optimization problem $F(\mathbf{x})$, a corresponding phase I optimization problem can be constructed as

$$F_I(\mathbf{x}) := \begin{cases} \text{minimize} & \rho \\ \text{subject to} & f_i(\mathbf{x}) \leq \rho, i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, j = 1, \ldots, t, \end{cases}$$

where $\rho$ is a single variable. It is obvious that when $\rho$ is large enough, $F_I(\mathbf{x})$ is always feasible.

Suppose $\mathbf{x}^*$ minimizes the objective function and $\rho^*$ is the corresponding minimum value. The phase I problem is designed in such a way that when $\rho^* \leq 0$, the original problem $F(\mathbf{x})$ is feasible and $F(\mathbf{x})$ is infeasible otherwise. Thus the verification scheme for infeasible case can be designed as follows. When the cloud indicates that the solutions to the two outsourced problem $G(\mathbf{y})$ and $H(\mathbf{z})$ are infeasible, it then generates the corresponding two phase I problems $G_I(\mathbf{y})$ and $H_I(\mathbf{z})$ and computes the optimal points $\mathbf{y}^*$ and $\mathbf{z}^*$ and the minimum values $\rho_G^*$ and $\rho_H^*$, respectively. Then at the local side, the verification is the same as that in the normal case. That is only when $\rho_G^* > 0$ and $\rho_H^* > 0$ and the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds can the end-user be guaranteed to receive valid solutions.

#### 6.2.3 Unbounded Case

In the unbounded case, the cloud indicates that the objective function $f_0(\mathbf{x}) \to -\infty$ in its domain. We utilize duality to verify the soundness of the returned result. For a convex optimization problem, we can construct the corresponding *Lagrangian* $L$ as

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} u_i f_i(\mathbf{x}) + \sum_{j=1}^{t} v_j h_j(\mathbf{x}),$$

where $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^t$ are the associated *Lagrange multiplier vectors*. Then based on this Lagrangian $L(\mathbf{x}, \mathbf{u}, \mathbf{v})$, a *Lagrange dual function* can be constructed as

$$\begin{aligned} \Phi(\mathbf{u}, \mathbf{v}) &= \inf_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \mathbf{u}, \mathbf{v}) \\ &= \inf_{\mathbf{x} \in \mathcal{D}} \left( f_0(\mathbf{x}) + \sum_{i=1}^{m} u_i f_i(\mathbf{x}) + \sum_{j=1}^{t} v_j h_j(\mathbf{x}) \right), \end{aligned}$$

where $\mathcal{D}$ is the domain of the optimization problem. From this definition, it is easy to prove that $\forall \mathbf{u} \succeq 0$, we have the following inequality:

$$\Phi(\mathbf{u}, \mathbf{v}) \leq L(\mathbf{x}^*, \mathbf{u}, \mathbf{v}) \leq f_0(\mathbf{x}^*),$$

where $f_0(\mathbf{x}^*)$ denotes the optimal value of the objective function. The above inequality gives a lower bound of the objective function that depends on the selection of $\mathbf{u}$ and $\mathbf{v}$. Thus, among all the selections of $\mathbf{u}$ and $\mathbf{v}$, finding the optimal lower bound is equivalent to solving the following optimization problem:

$$\begin{cases} \text{maximize} & \Phi(\mathbf{u}, \mathbf{v}) \\ \text{subject to} & \mathbf{u} \succeq 0. \end{cases}$$

The objective function $\Phi(\mathbf{u}, \mathbf{v})$ is concave since it is the point-wise infimum of a series of affine function of $(\mathbf{u}, \mathbf{v})$. Thus the above optimization problem is also a convex optimization problem. If the original problem is unbounded below, the convex optimization problem described above should be infeasible since it gives a lower bound of the optimal value in the original problem. Thus the remaining task is to verify the feasibility of the above convex optimization problem, which has been illustrated in the infeasible case. Let the cloud solve the phase I problems of the two Lagrange dual problems and return the optimal solutions denoted by $(\rho_G^*, \mathbf{y}^*, \mathbf{u}_G^*, \mathbf{v}_G^*)$ and $(\rho_H^*, \mathbf{z}^*, \mathbf{u}_H^*, \mathbf{v}_H^*)$. At the local side, the end-user then checks whether $\rho_G^* > 0$ and $\rho_H^* > 0$ and whether the equality $\mathbf{K}_1\mathbf{y}^* + \mathbf{r}_1 = \mathbf{K}_2\mathbf{z}^* + \mathbf{r}_2$ holds.

## 7 Evaluation

In this section, we will evaluate the performance of the proposed CASO scheme. We first compare CASO with several

existing outsourcing schemes. Then we present some numeric results to show the efficiency of CASO.

### 7.1 Performance Comparison

The existing schemes on outsourcing of numeric computation mainly focus on some specific problems. To the best of our knowledge, no effective outsourcing schemes have been proposed for non-linear problems. In the following part, we compare the performance of our proposed CASO scheme with three existing schemes specially designed for three types of problems in terms of security, computational complexity and communication overhead. To measure the communication overhead, we introduce a *communication overhead index* $\mathcal{I}_c$ which is defined as the fraction of the communication cost of transmitting the original problem over that of the transformed problem. Thus a larger $\mathcal{I}_c$ indicates better communication efficiency.

#### 7.1.1 Linear Programming

In this section, we compare CASO for linear programming problems with the schemes proposed in [6] and [23] in both security and complexity. We will show that while achieving the same security level, our scheme outperforms them in terms of complexity. In addition, our scheme also provides end-users with the flexibility to select different outsourcing options with different complexity according to their security demands.

The general linear programming problems can be expressed as

$$\begin{cases} \text{minimize} & \mathbf{c^T x} \\ \text{subject to} & \mathbf{Ax = b} \\ & \mathbf{Dx \geq 0}. \end{cases} \tag{11}$$

In [6], to transform the problem, a secret key $\mathbf{K} = \{\mathbf{Q}, \mathbf{M}, \mathbf{r}, \lambda, \gamma\}$ is generated, where $\mathbf{Q}$ is a randomly generated $m \times m$ non-singular matrix, $\mathbf{M}$ is a randomly generated $n \times n$ non-singular matrix, and $\mathbf{r}$ is an $n \times 1$ vector. With this secret key, the original problem is transformed to the following problem

$$\begin{cases} \text{minimize} & \mathbf{c'^T x} \\ \text{subject to} & \mathbf{A'x = b'} \\ & \mathbf{D'x \geq 0}, \end{cases} \tag{12}$$

where $\mathbf{A'} = \mathbf{QAM}, \mathbf{D'} = (\mathbf{D} - \lambda\mathbf{QA})\mathbf{M}, \mathbf{b'} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar})$ and $\mathbf{c'} = \gamma\mathbf{M^T c}$. Then the transformed problem is outsourced to the cloud which is similar as our approach.

In terms of computational complexity, the computational overhead of the outsourcing scheme in [6] as well as our scheme lies primarily in matrix multiplication. As stated in their paper, the overall computational complexity for the scheme proposed in [6] is slightly less than $\mathcal{O}(n^3)$ depending the algorithm chosen to implement matrix multiplication. For instance, when the Strassen algorithm is adopted, the complexity becomes $\mathcal{O}(n^{2.81})$; while for the Coppersmith-Winograd algorithm the complexity is $\mathcal{O}(n^{2.376})$. However, by carefully selecting the secret key $\mathbf{K}$, our scheme can limit the complexity within $\mathcal{O}(n^2)$.

In terms of communication overhead, the original problems in both schemes are transformed by matrix multiplication such that the resulting matrices are still in the same scale. As a result, the communication cost of the original and transformed problems are in the same level. Thus we have $\mathcal{I}_c = 1$ in our scheme and the scheme in [6].

In terms of security, both schemes can conceal the private information by some disguising techniques, that is to disguise the original matrices by multiplying them with some random matrices. As a consequence, the security they can achieve in protecting the original coefficient matrix is in the same level. Since the types of the transformation matrices (e.g., $\mathbf{Q}$, $\mathbf{M}$) are not specified, each entry in the disguised coefficient matrix $\mathbf{A'}$ can be the linear combination of multiple entries in $\mathbf{A}$ and the transformation matrices. Thus, the ratio information can be concealed. In this sense, the security of the scheme in [6] is comparable with our scheme-4 in terms of protecting side information.

The scheme proposed in [23] can be regarded as a variation of that in [6]. The main difference is that the authors in [23] specify the transformation matrices as sparse matrices in order to achieve a lower computational complexity of $\mathcal{O}(n^2)$. For example, the schemes in [23] disguises the coefficient matrix by matrix multiplication as $\mathbf{A'} = \mathbf{MAN}$, where $\mathbf{M}$ and $\mathbf{N}$ are both sparse matrices. In this way, the complexity is reduced to $\mathcal{O}(n^2)$. Actually, this scheme can be considered as a special case of our proposed CASO where $\mathbf{K}$ is selected as a sparse matrix.

#### 7.1.2 System of Linear Equations

In [24], the authors investigated outsourcing of system of linear equations $\mathbf{Ax = b}$ based on iterative method. First, the problem is transformed to $\mathbf{Ay = b'}$, where $\mathbf{y = x + r}$, $\mathbf{b' = b + Ar}$ and $\mathbf{r}$ is a random vector. Then the end-user solves the transformed problem iteratively with the aid of cloud servers and an initial guess $\mathbf{y}_0$ from the following iteration equation:

$$\mathbf{y_{k+1}} = \mathbf{T} \cdot \mathbf{y_k} + \mathbf{c'}, \tag{13}$$

where $\mathbf{A = D + R}$ such that $\mathbf{D}$ is non-singular, $\mathbf{T} = -\mathbf{D^{-1}} \cdot \mathbf{R}$ and $\mathbf{c'} = \mathbf{D^{-1}} \cdot \mathbf{b'}$. The end-user utilizes the cloud servers to compute the most expensive part $\mathbf{T} \cdot \mathbf{y_k}$ based on homomorphic encryption to conceal the private information $\mathbf{T}$. To be specific, the matrix $\mathbf{T}$ is pre-computed at the local side and the encrypted version $\mathsf{Enc}(\mathbf{T})$ is outsourced to the cloud. At each iteration, the end-user sends $\mathbf{y_k}$ to the cloud and based on the homomorphic properties of the encryption, the cloud servers compute $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ by

$$\begin{aligned} \mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})[i] &= \mathsf{Enc}\left(\sum_{j=1}^{n} \mathbf{T}[i,j] \cdot y_{k,j}\right) \\ &= \prod_{j=1}^{n} \mathsf{Enc}(\mathbf{T}[i,j])^{y_{k,j}}, \end{aligned}$$

for $i = 1, \ldots, n$ and send $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ back to the end-user. On receiving $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$, the end-user decrypts it and get $\mathbf{y_{k+1}}$. This iteration terminates when it converges to the final result $\mathbf{y}$. At last the end-user can recover the desired solution $\mathbf{x}$ by $\mathbf{x = y - r}$.

As stated above, the computational overhead at the local side primarily lies in the decryption of $\mathbf{T} \cdot \mathbf{y_k}$ in each iteration. Suppose the algorithm terminates after $l$ rounds of iteration, then the end-user has to perform $l \cdot n$ times of

TABLE 4
Performance Comparison

| | Applicability | | | | Computational Complexity | Communication Overhead Index $\mathcal{I}_c$ |
|---|---|---|---|---|---|---|
| | LE | LP | NLE | COPT | | |
| Our Scheme | √ | √ | √ | √ | $\mathcal{O}(n^2)$ | 1 |
| [6] | | √ | | | $\mathcal{O}(n^{2.376})$ | 1 |
| [24] | √ | | | | $\mathcal{O}(l \cdot n^{3+\epsilon})$ | $\frac{1}{l}$ |
| [29] | | | | Only Verification | Not Applicable | Not Applicable |

decryption. However, the decryption process of public-key cryptosystem is much more expensive than simple multiplication of real numbers since it mainly consists of modular exponentiation of large numbers. For instance, the decryption process [30] adopted in [24] has a complexity of $\mathcal{O}(n^3)$ and a modified version can achieve a complexity of $\mathcal{O}(n^{2+\epsilon})$. Thus, the outsourcing scheme in [24] introduces $\mathcal{O}(n^{3+\epsilon})$ computational overhead at the local side. In terms of communication overhead, the outsourcing process requires the end-user to send $\mathbf{y_k}$ and receive $\mathsf{Enc}(\mathbf{T} \cdot \mathbf{y_k})$ at each iteration. As a consequence, the communication overhead index $\mathcal{I}_c = \frac{1}{l}$ is dependent on the convergence speed. Furthermore, this iteration process requires the end-user to be "online" for the process to continue. In comparison, our scheme can limit the computational overhead to $\mathcal{O}(n^2)$ with $\mathcal{I}_c = 1$. Moreover, during the outsourcing process, the end-user is "offline", which means that after outsourcing the transformed problem, the end-user does not need to interact with the cloud servers until the result is sent back.

The system of linear equations considered in [24] includes the coefficient matrix $\mathbf{T}$ and the solution vector $\mathbf{x}$. In [24], the matrix $\mathbf{T}$ is encrypted utilizing the Paillier cryptosystem [30] as $\mathsf{Enc}(\mathbf{T})$ and the vector $\mathbf{x}$ is transformed to $\mathbf{y} = \mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is a random vector. In comparison, CASO disguises the coefficient matrix $\mathbf{A}$ and the solution vector $\mathbf{x}$ as $\mathbf{A}' = \mathbf{AK}$ and $\mathbf{x} = \mathbf{Ky} + \mathbf{r}$, respectively. In the Paillier cryptosystem, each entry of the coefficient matrix $\mathbf{T}(i,j)$ is encrypted as $\mathsf{Enc}(\mathbf{T}(i,j)) = g^{\mathbf{T}(i,j)} r^n \bmod n^2$, where $g, r, n$ are parameters in the cryptosystem. There are two scenarios: (i) If $r$'s are the same for all entries in the coefficient matrix, then all the identical entries in $\mathbf{A}$ will be encrypted to identical entries in $\mathbf{A}'$. In other words, by inspecting identical entries in $\mathbf{A}'$, we can determine whether entries in $\mathbf{A}$ are identical or not. However, in CASO, since an entry in $\mathbf{A}'$ is the linear combination of entries in $\mathbf{A}$ and $\mathbf{K}$, the identical entries in $\mathbf{A}'$ would not indicate that the corresponding entries in $\mathbf{A}$ are identical. Thus, in this case, CASO will *provide better security protection*. In this case, the end-user needs to compute $n^2 + 1$ exponential operation. (ii) If a different $r$ is used for each entry of the coefficient matrix, then the end-user has to randomly select $n^2$ $r$'s, which is quite complex. Furthermore, the end-user need to compute 2 exponential operations for each entry ($g^{a_{i,j}}$ and $r^n$). Therefore, altogether, the end-user has to compute $2n^2$ exponential operations. In addition, due to security requirement, $n$ has to be at least 1,024 bits long. In this case, $n^2$ would be 2048 bits. As an example, the size of the outsourced coefficient matrix for 5,000 variables would be around 6 MB without data compression. While in scheme-1 and scheme-2 of our proposed CASO, the transformation is applied in the column basis. As a result, the order information of each column may be exposed. In this sense, the scheme in [24] may provide better protection than scheme-1 and scheme-2 regarding the coefficient matrix $\mathbf{A}$. However, in scheme-3, each entry in $\mathbf{A}$ is transformed to

$$a'_{ij} = \sum_{r=j-\omega}^{j+\omega} a_{ir} k_{rj}.$$

When $\omega > 0$, since each $k_{rj}$ in $\mathbf{K}$ is randomly chosen, the order information in each column will also be concealed. Thus the scheme in [24] can provide comparable security protection regrading the coefficient matrix $\mathbf{A}$ as scheme-3. In scheme-4, the entries in $\mathbf{A}'$ are further permuted. As a result, there exist no explicit relation between the entry $a_{ij}$ in $\mathbf{A}$ and the corresponding entry $a'_{ij}$ in $\mathbf{A}'$. However, one can know for sure that the entry $t'_{ij}$ in $\mathsf{Enc}(\mathbf{T})$ is encrypted from the entry $t_{ij}$ in $\mathbf{T}$. Thus scheme-4 can provide better protection of $\mathbf{A}$.

In terms of the solution vector $\mathbf{x}$, in [24], the solution vector $\mathbf{x}$ is protected by adding a random vector $\mathbf{r}$ as $\mathbf{y} = \mathbf{x} + \mathbf{r}$, while in our scheme, we conceal $\mathbf{x}$ by the affine mapping $\mathbf{x} = \mathbf{Ky} + \mathbf{r}$. Thus, CASO scheme can provide better security protection in this aspect.

#### 7.1.3 Convex Optimization

In [29], the authors proposed a verification scheme for convex optimization problems. However, they did not give any outsourcing scheme. Compare to [29], in addition to result verification, CASO also provides a secure outsourcing scheme. Even in result verification, CASO outperforms it in terms of computational complexity.

The result verification of convex optimization is divided into three categories: normal, infeasible and unbounded. The verification for normal case forms the basis for other two cases. For the normal case, the basic idea in [29] is to check the Karush-Kuhn-Tucker (KKT) optimality condition. The end-user has to evaluate the original functions as well as their differentials based on the optimal points returned by the cloud. This verification process is much more expensive since all the original functions are non-linear. In comparison, our verification scheme requires only linear operations (e.g., multiplication and addition) on the independent variables and the returned solution, therefore, it must be more efficient.

#### 7.1.4 Summary

We summarize the performance comparison of CASO with some existing works in Table 4. We have shown that in the case of outsourcing linear programming (LP) and system of linear equations (LE), CASO outperforms the existing schemes in computational complexity. In terms of security, all the schemes are secure in protecting the original

TABLE 5
Performance Evaluation for System of Linear Equations

| Dimension | Bandwidth | $\mathcal{T}_e$ (s) | $\mathcal{T}_s$ (s) | $\mathcal{I}$ |
|---|---|---|---|---|
| $n = 1000$ | $W = 1$ | 0.0265 | 0.2356 | 8.9 |
| | $W = 7$ | 0.0265 | 0.2402 | 9.1 |
| | $W = 15$ | 0.0546 | 0.2356 | 4.3 |
| | $W = 31$ | 0.0858 | 0.2387 | 2.8 |
| $n = 2000$ | $W = 1$ | 0.0593 | 1.3962 | 23.6 |
| | $W = 7$ | 0.0936 | 1.4071 | 15.0 |
| | $W = 15$ | 0.1248 | 1.3853 | 11.1 |
| | $W = 31$ | 0.1950 | 1.3494 | 6.9 |
| $n = 3000$ | $W = 1$ | 0.1170 | 3.9234 | 33.5 |
| | $W = 7$ | 0.1856 | 3.9281 | 21.2 |
| | $W = 15$ | 0.3058 | 3.8844 | 12.7 |
| | $W = 31$ | 0.4867 | 3.8766 | 8.0 |
| $n = 4000$ | $W = 1$ | 0.2184 | 8.5832 | 39.3 |
| | $W = 7$ | 0.3416 | 8.6924 | 25.4 |
| | $W = 15$ | 0.7129 | 8.6565 | 12.1 |
| | $W = 31$ | 1.0171 | 8.6768 | 8.5 |
| $n = 5000$ | $W = 1$ | 0.3260 | 15.8138 | 48.5 |
| | $W = 7$ | 0.5288 | 15.9839 | 30.2 |
| | $W = 15$ | 1.2683 | 15.8793 | 12.5 |
| | $W = 31$ | 1.8174 | 15.9698 | 8.8 |

coefficient matrix. That is, given the disguised problem, input and output, it is computational infeasible to recover the original problem, input and output. CASO can also be applied to system of non-linear equations (NLE) and convex optimization (COPT). This shows that CASO possesses better applicability. Furthermore, compared to the existing works, CASO also gives end-users the flexibility to choose the most suitable outsourcing strategy on a cost-aware basis. That is the end user can select the secret key $\mathbf{K}$ for the outsourcing scheme based on its various security demands and computational resources.

### 7.2 Numeric Results

In this section, we measure the performance of CASO utilizing MATLAB. The computation of both the end-user and the cloud server is simulated using the same computer with an Intel Core 2 Due CPU running at 2. 53 GHz with 4 GB RAM. We take outsourcing of the system of linear and non-linear equations as examples. In the process of outsourcing, we focus on the overhead of problem transformation, result recovery and the performance gain that they can achieve by outsourcing problems to the cloud. We denote the time for local computation in the outsourcing process $\mathcal{T}_e$, the time cost without outsourcing $\mathcal{T}_s$, and the performance gain $\mathcal{I} = \mathcal{T}_s/\mathcal{T}_e$.

We first show the simulation results for outsourcing of system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ matrix. In complexity analysis, we show that the complexities of scheme-1 and scheme-2 are in the same level while the complexity for scheme-3 and scheme-4 are comparable.

In scheme-3, when the bandwidth $W$ equals 1, it is reduced to scheme-1. Thus in our evaluation, we take scheme-3 as an example and let $\mathbf{K}$ be a band matrix with bandwidth $W$ varying from 1 to 31. To investigate the impact of problem size on our proposed scheme, we let $n$ vary from 1,000 to 5,000. The numeric results are shown in Table 5. First, we can learn from the results that when the bandwidth of the banded matrix $\mathbf{K}$ becomes larger, the computational overhead at local side grows and the performance gain decreases. This fact coincides with our analysis of the trade off between complexity and security. Second, the performance gain increases with the growth of the problem dimension $n$. This is because our scheme requires the end-users to carry out simple operations such as addition and multiplication. And this feature becomes more obvious for the case of non-linear computation.

TABLE 6
Performance Evaluation for System of Non-Linear Equations

| Dimension | Bandwidth | $\mathcal{T}_e$ (s) | $\mathcal{T}_s$(s) | $\mathcal{I}$ |
|---|---|---|---|---|
| $n = 1000$ | $W = 1$ | 1.6800 | 26.2 | 15.6 |
| | $W = 2$ | 2.4500 | 27.1 | 11.1 |
| | $W = 3$ | 3.0500 | 26.2 | 8.6 |
| $n = 2000$ | $W = 1$ | 3.1500 | 118.2 | 37.5 |
| | $W = 2$ | 5.1200 | 118.8 | 23.2 |
| | $W = 3$ | 6.3900 | 117.1 | 18.3 |
| $n = 3000$ | $W = 1$ | 5.1300 | 330.8 | 64.5 |
| | $W = 2$ | 7.7100 | 313.0 | 40.6 |
| | $W = 3$ | 9.7500 | 320.6 | 32.9 |
| $n = 4000$ | $W = 1$ | 7.1600 | 712.9 | 99.6 |
| | $W = 2$ | 12.3800 | 713.1 | 57.6 |
| | $W = 3$ | 13.9000 | 711.4 | 51.2 |
| $n = 5000$ | $W = 1$ | 9.3700 | 1187.2 | 126.7 |
| | $W = 2$ | 16.0000 | 1190.1 | 74.4 |
| | $W = 3$ | 20.6700 | 1191.1 | 57.6 |

Then we show the performance of our proposed scheme for system of non-linear equations. We assume that the non-linear system is composed of polynomials on ten variables and let the number of independent terms $N$ vary from 1,000 to 5,000. Also for the same reason, we deploy band matrix as the key matrix and let the bandwidth $W$ vary from 1 to 3. The simulation result is shown in Table 6. For system of non-linear equations, the performance gain is larger than its linear counterpart. This is because CASO requires only linear operations (e.g., multiplication and addition) in the local environment. Similar to that of the system of linear equations, the results clearly show that there exists a trade-off between the computational complexity and security.

## 8 Conclusion

In this paper, we proposed a cost-aware secure outsourcing scheme for general computational problems. We demonstrated that CASO can be utilized for secure outsourcing of various computational problems, such as system of equations, linear programming and convex optimizations. Our scheme also provides mechanisms for the end-users to verify results received from the cloud. We provided security analysis on our proposed scheme on a cost-aware basis. In particular, we proved that CASO is secure in protecting the coefficient matrix of the outsourced problem and can partly conceal the side information. Our analysis shows that CASO can limit the computational overhead at the local side to $\mathcal{O}(n^2)$. Since CASO is executed off-line, the communication overhead is in the same level as that of outsourcing the original problem itself. We also compared CASO with several existing schemes and showed that CASO is more efficient and has a wider applicability.

## ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under CNS:1524520.

## REFERENCES

[1] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Secur.*, vol. 4, no. 4, pp. 277–287, 2005.
[2] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2012, pp. 505–522.
[3] M. Blanton and M. Aliasgari, "Secure outsourcing of DNA searching via finite automata," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2010, pp. 49–64.
[4] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf. Comput. Commun. Secur.*, 2010, pp. 48–59.
[5] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. 6th Annu. Conf. Privacy Secur. Trust*, 2008, pp. 240–245.
[6] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. IEEE INFOCOM*, 2011, pp. 820–828.
[7] Y. N. Seitkulov, "New methods of secure outsourcing of scientific computations," *J. Supercomput.*, vol. 65, no. 1, pp. 469–482, 2013.
[8] K. Zhou and J. Ren, "LinSOS: Secure outsourcing of linear computations based on affine mapping," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–5.
[9] V. S. Rao and N. Satyanarayana, "Secure and practical outsourcing of linear programming in cloud computing: A survey," *Int. J. Comput. Appl.*, vol. 159, no. 4, pp. 1–4, 2017.
[10] W. Shen, B. Yin, X. Cao, Y. Cheng, and X. S. Shen, "A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, p. 1, Jan. 2017, doi: 10.1109/TCC.2016.2647718.
[11] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. Theory Cryptography Conf.*, 2005, pp. 264–282.
[12] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.
[13] K. Zhou, M. H. Afifi, and J. Ren, "ExpSOS: Secure and verifiable outsourcing of exponentiation operations for mobile cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2518–2531, Nov. 2017.
[14] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2401–2414, Nov. 2016.
[15] L. Li, R. Lu, K.-K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on vertically partitioned databases," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 8, pp. 1847–1861, Aug. 2016.
[16] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3411–3425, Jul. 2016.
[17] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.
[18] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 492–503.
[19] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. thesis, Stanford University, Stanford, CA, 2009.
[20] A. C. Yao, "Protocols for secure computations," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, 1982, pp. 160–164.
[21] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. Advances Cryptology Conf.*, 2010, pp. 465–482.
[22] M. J. Atallah, K. N. Pantazopoulos, J. Rice, and E. Spafford, "Secure outsourcing of scientific computations," *Advances Comput.*, vol. 54, pp. 215–272, 2002.
[23] H. Nie, X. Chen, J. Li, J. Liu, and W. Lou, "Efficient and verifiable algorithm for secure outsourcing of large-scale linear programming," in *Proc. IEEE 28th Int. Conf. Advanced Inf. Netw. Appl.*, 2014, pp. 591–596.
[24] C. Wang, K. Ren, J. Wang, and K. M. Urs, "Harnessing the cloud for securely solving large-scale systems of linear equations," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 549–558.
[25] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 1, pp. 69–78, Jan. 2015.
[26] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.
[27] S. Pissanetzky, *Sparse Matrix Technology*. Cambridge, MA, USA: Academic Press, 1984.
[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[29] Z. Xu, C. Wang, Q. Wang, K. Ren, and L. Wang, "Proof-carrying cloud computation: The case of convex optimization," in *Proc. IEEE INFOCOM*, 2013, pp. 610–614.
[30] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

**Kai Zhou** (S'16) received the BS degree in electrical engineering from Shanghai Jiao Tong University, China, in 2013. He is working towards the PhD degree in electrical and computer engineering at Michigan State University. His research interests include applied cryptography, cloud security and privacy, coding theory, and secure communication. He is a student member of the IEEE.

**Jian Ren** (SM'09) received the BS and MS degrees both in mathematics from Shaanxi Normal University, and the PhD degree in EE from Xidian University, China. He is an associate professor with the Department of ECE, Michigan State University. His current research interests include network security, cloud computing security, privacy-preserving communications, distributed network storage, and internet of things. He is a recipient of the US National Science Foundation Faculty Early Career Development (CAREER) award in 2009. He is the TPC chair of IEEE ICNC'17 and general chair of ICNC'18. He is a senior member of the IEEE.