This paper was presented as part of the main technical program at IEEE INFOCOM 2011

Secure and Practical Outsourcing of Linear Programming in Cloud Computing

Cong Wang, Kui Ren, and Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL 60616, USA
Email: {cong, kren, jwang}@ece.iit.edu

***Abstract*—Cloud computing enables customers with limited computational resources to outsource large-scale computational tasks to the cloud, where massive computational power can be easily utilized in a pay-per-use manner. However, security is the major concern that prevents the wide adoption of computation outsourcing in the cloud, especially when end-user's confidential data are processed and produced during the computation. Thus, secure outsourcing mechanisms are in great need to not only protect sensitive information by enabling computations with encrypted data, but also protect customers from malicious behaviors by validating the computation result. Such a mechanism of general secure computation outsourcing was recently shown to be feasible in theory, but to design mechanisms that are practically efficient remains a very challenging problem.**

Focusing on engineering computing and optimization tasks, this paper investigates secure outsourcing of widely applicable linear programming (LP) computations. In order to achieve practical efficiency, our mechanism design explicitly decomposes the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The resulting flexibility allows us to explore appropriate security/efficiency tradeoff via higher-level abstraction of LP computations than the general circuit representation. In particular, by formulating private data owned by the customer for LP problem as a set of matrices and vectors, we are able to develop a set of efficient privacy-preserving problem transformation techniques, which allow customers to transform original LP problem into some random one while protecting sensitive input/output information. To validate the computation result, we further explore the fundamental duality theorem of LP computation and derive the necessary and sufficient conditions that correct result must satisfy. Such result verification mechanism is extremely efficient and incurs close-to-zero additional cost on both cloud server and customers. Extensive security analysis and experiment results show the immediate practicability of our mechanism design.

I. Introduction

Cloud Computing provides convenient on-demand network access to a shared pool of configurable computing resources that can be rapidly deployed with great efficiency and minimal management overhead [1]. One fundamental advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constraint devices. By outsourcing the workloads into the cloud, customers could enjoy the literally unlimited computing resources in a pay-per-use manner without committing any large capital outlays in the purchase of both hardware and software and/or the operational overhead therein.

Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customers' direct control over the systems that consume and produce their data during the computation, which inevitably brings in new security concerns and challenges towards this promising computing model [2]. On the one hand, the outsourced computation workloads often contain sensitive information, such as the business financial records, proprietary research data, or personally identifiable health information etc. To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing [2] so as to provide end-to-end data confidentiality assurance in the cloud and beyond. However, ordinary data encryption techniques in essence prevent cloud from performing any meaningful operation of the underlying plaintext data [3], making the computation over encrypted data a very hard problem. On the other hand, the operational details inside the cloud are not transparent enough to customers [4]. As a result, there do exist various motivations for cloud server to behave unfaithfully and to return incorrect results, i.e., they may behave beyond the classical semi-honest model. For example, for the computations that require a large amount of computing resources, there are huge financial incentives for the cloud to be "lazy" if the customers cannot tell the correctness of the output. Besides, possible software bugs, hardware failures, or even outsider attacks might also affect the quality of the computed results. Thus, we argue that the cloud is intrinsically *not secure* from the viewpoint of customers. Without providing a mechanism for secure computation outsourcing, i.e., to protect the sensitive input and output information of the workloads and to validate the integrity of the computation result, it would be hard to expect cloud customers to turn over control of their workloads from local machines to cloud solely based on its economic savings and resource flexibility. For practical consideration, such a design should further ensure that customers perform less amount of operations following the mechanism than completing the computations by themselves directly. Otherwise, there is no point for customers to seek help from cloud.

Recent researches in both the cryptography and the theoretical computer science communities have made steady advances in "secure outsourcing expensive computations" (e.g. [5]–[10]). Based on Yao's garbled circuits [11] and Gentry's breakthrough work on fully homomorphic encryption (FHE) scheme [12], a general result of secure computation outsourcing has been shown viable in theory [9], where the computation is represented by an encrypted combinational boolean

978-1-4244-9921-2/11/$26.00 ©2011 IEEE

circuit that allows to be evaluated with encrypted private inputs. However, applying this general mechanism to our daily computations would be far from practical, due to the extremely high complexity of FHE operation as well as the pessimistic circuit sizes that cannot be handled in practice when constructing original and encrypted circuits. This overhead in general solutions motivates us to seek efficient solutions at higher abstraction levels than the circuit representations for specific computation outsourcing problems. Although some elegant designs on secure outsourcing of scientific computations, sequence comparisons, and matrix multiplication etc. have been proposed in the literature, it is still hardly possible to apply them directly in a practically efficient manner, especially for large problems. In those approaches, either heavy cloud-side cryptographic computations [7], [8], or multi-round interactive protocol executions [5], or huge communication complexities [10], are involved (detailed discussions in Section VI). In short, practically efficient mechanisms with immediate practices for secure computation outsourcing in cloud are still missing.

Focusing on engineering computing and optimization tasks, in this paper, we study practically efficient mechanisms for secure outsourcing of linear programming (LP) computations. Linear programming is an algorithmic and computational tool which captures the first order effects of various system parameters that should be optimized, and is essential to engineering optimization. It has been widely used in various engineering disciplines that analyze and optimize real-world systems, such as packet routing, flow control, power management of data centers, etc. [13]. Because LP computations require a substantial amount of computational power and usually involve confidential data, we propose to explicitly decompose the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The flexibility of such a decomposition allows us to explore higher-level abstraction of LP computations than the general circuit representation for the practical efficiency.

Specifically, we first formulate private data owned by the customer for LP problem as a set of matrices and vectors. This higher level representation allows us to apply a set of efficient privacy-preserving problem transformation techniques, including matrix multiplication and affine mapping, to transform the original LP problem into some arbitrary one while protecting the sensitive input/output information. One crucial benefit of this higher level problem transformation method is that existing algorithms and tools for LP solvers can be directly reused by the cloud server. Although the generic mechanism defined at circuit level, e.g. [9], can even allow the customer to hide the fact that the outsourced computation is LP, we believe imposing this more stringent security measure than necessary would greatly affect the efficiency. To validate the computation result, we utilize the fact that the result is from cloud server solving the transformed LP problem. In particular, we explore the fundamental duality theorem together with the piece-wise construction of auxiliary LP problem to derive a set of necessary and sufficient conditions that the correct result must satisfy. Such a method of result validation can be very efficient and incurs close-to-zero additional overhead on both customer and cloud server. With correctly verified result, customer can use the secret transformation to map back the desired solution for his original LP problem. We summarize our contributions as follows:

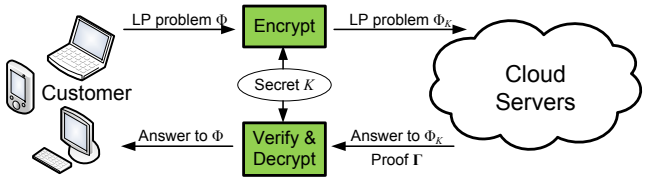

Fig. 1: Architecture of secure outsourcing linear programming problems in Cloud Computing

1) For the first time, we formalize the problem of securely outsourcing LP computations, and provide such a secure and practical mechanism design which fulfills input/output privacy, cheating resilience, and efficiency.
2) Our mechanism brings cloud customer great computation savings from secure LP outsourcing as it only incurs $O(n^\rho)$ for some $2 < \rho \leq 3$ local computation overhead on the customer, while solving a normal LP problem usually requires more than $O(n^3)$ time [13].
3) The computations done by the cloud server shares the same time complexity of currently practical algorithms for solving the linear programming problems, which ensures that the use of cloud is economically viable.
4) The experiment evaluation further demonstrates the immediate practicality: our mechanism can always help customers achieve more than $30\times$ savings when the sizes of the original LP problems are not too small, while introducing no substantial overhead on the cloud.

The rest of the paper is organized as follows. Section II introduces the system and threat model, and our design goals. Then we provide the detailed mechanism description in Section III. Section IV and V give the security analysis and performance evaluation, followed by Section VI which overviews the related work. Finally, Section VII gives the concluding remark of the whole paper.

II. Problem Statement

A. System and Threat Model

We consider a computation outsourcing architecture involving two different entities, as illustrated in Fig. 1: the *cloud customer*, who has large amount of computationally expensive LP problems to be outsourced to the cloud; the *cloud server* (CS), which has significant computation resources and provides utility computing services, such as hosting the public LP solvers in a pay-per-use manner.

The customer has a large-scale linear programming problem Φ (to be formally defined later) to be solved. However, due to the lack of computing resources, like processing power, memory, and storage etc., he cannot carry out such expensive computation locally. Thus, the customer resorts to CS for solving the LP computation and leverages its computation

capacity in a pay-per-use manner. Instead of directly sending original problem Φ, the customer first uses a secret K to map Φ into some encrypted version Φ_K and outsources problem Φ_K to CS. CS then uses its public LP solver to get the answer of Φ_K and provides a correctness proof Γ, but it is supposed to learn nothing or little of the sensitive information contained in the original problem description Φ. After receiving the solution of encrypted problem Φ_K, the customer should be able to first verify the answer via the appended proof Γ. If it's correct, he then uses the secret K to map the output into the desired answer for the original problem Φ.

The security threats faced by the computation model primarily come from the malicious behavior of CS. We assume that the CS may behave beyond "honest-but-curious", i.e. the semi-honest model that was assumed by many previous researches (e.g., [14], [15]), either because it intends to do so or because it is compromised. The CS may be persistently interested in analyzing the encrypted input sent by the customer and the encrypted output produced by the computation to learn the sensitive information as in the semi-honest model. In addition, CS can also behave unfaithfully or intentionally sabotage the computation, e.g. to lie about the result to save the computing resources, while hoping not to be caught at the same time.

We assume the communication channels between the cloud server and the customer are reliably authenticated, which can be achieved in practice with little overhead. These authentication handshakes are omitted in the following presentation.

B. Design Goals

To enable secure and practical outsourcing of LP under the aforementioned model, our mechanism design should achieve the following security and performance guarantees.

1) **Correctness**: Any cloud server that faithfully follows the mechanism must produce an output that can be decrypted and verified successfully by the customer.
2) **Soundness**: No cloud server can generate an incorrect output that can be decrypted and verified successfully by the customer with non-negligible probability.
3) **Input/output privacy**: No sensitive information from the customer's private data can be derived by the cloud server during performing the LP computation.
4) **Efficiency**: The local computations done by customer should be substantially less than solving the original LP on his own. The computation burden on the cloud server should be within the comparable time complexity of existing practical algorithms solving LP problems.

C. Background on Linear Programming

An optimization problem is usually formulated as a mathematical programming problem that seeks the values for a set of decision variables to minimize (or maximize) an objective function representing the cost subject to a set of constraints. For linear programming, the objective function is an affine function of the decision variables, and the constraints are a system of linear equations and inequalities. Since a constraint in the form of a linear inequality can be expressed as a linear equation by introducing a non-negative slack variable, and a free decision variable can be expressed as the difference of two non-negative auxiliary variables, any linear programming problem can be expressed in the following standard form,

$$\text{minimize} \quad \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}. \qquad (1)$$

Here $\mathbf{x}$ is an $n \times 1$ vector of decision variables, $\mathbf{A}$ is an $m \times n$ matrix, and both $\mathbf{c}$ and $\mathbf{b}$ are $n \times 1$ vectors. It can be assumed further that $m \leq n$ and that $\mathbf{A}$ has full row rank; otherwise, extras rows can always be eliminated from $\mathbf{A}$.

In this paper, we study a more general form as follows,

$$\text{minimize} \quad \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{B}\mathbf{x} \geq \mathbf{0}. \qquad (2)$$

In Eq. (2), we replace the non-negative requirements in Eq. (1) by requiring that each component of $\mathbf{B}\mathbf{x}$ to be non-negative, where $\mathbf{B}$ is an $n \times n$ non-singular matrix, i.e. Eq. (2) degenerates to Eq. (1) when $\mathbf{B}$ is the identity matrix. Thus, the LP problem can be defined via the tuple $\Phi = (\mathbf{A}, \mathbf{B}, \mathbf{b}, \mathbf{c})$ as input, and the solution $\mathbf{x}$ as output.

III. The Proposed Schemes

This section presents our LP outsourcing scheme which provides a *complete outsourcing* solution for – not only the privacy protection of problem input/output, but also its efficient result checking. We start from an overview of secure LP outsourcing design framework and discuss a few basic techniques and their demerits, which leads to a stronger problem transformation design utilizing affine mapping. We then discuss effective result verification by leveraging the duality property of LP. Finally, we give the full scheme description.

A. Mechanism Design Framework

We propose to apply problem transformation for mechanism design. The general framework is adopted from a generic approach [9], while our instantiation is completely different and novel. In this framework, the process on cloud server can be represented by algorithm ProofGen and the process on customer can be organized into three algorithms (KeyGen, ProbEnc, ResultDec). These four algorithms are summarized below and will be instantiated later.

- KeyGen$(1^k) \rightarrow \{K\}$. *This is a randomized key generation algorithm which takes a system security parameter k, and returns a secret key K that is used later by customer to encrypt the target LP problem.*
- ProbEnc$(K, \Phi) \rightarrow \{\Phi_K\}$. *This algorithm encrypts the input tuple Φ into Φ_K with the secret key K. According to problem transformation, the encrypted input Φ_K has the same form as Φ, and thus defines the problem to be solved in the cloud.*
- ProofGen$(\Phi_K) \rightarrow \{(\mathbf{y}, \Gamma)\}$. *This algorithm augments a generic solver that solves the problem Φ_K to produce both the output $\mathbf{y}$ and a proof Γ. The output $\mathbf{y}$ later decrypts to $\mathbf{x}$, and Γ is used later by the customer to verify the correctness of $\mathbf{y}$ or $\mathbf{x}$.*
- ResultDec$(K, \Phi, \mathbf{y}, \Gamma) \rightarrow \{\mathbf{x}, \perp\}$. *This algorithm may choose to verify either $\mathbf{y}$ or $\mathbf{x}$ via the proof Γ. In any case,*

a correct output $\mathbf{x}$ *is produced by decrypting* $\mathbf{y}$ *using the secret* K*. The algorithm outputs* $\perp$ *when the validation fails, indicating the cloud server was not performing the computation faithfully.*

Note that our proposed mechanism provides us one-time-pad type of flexibility. Namely, we shall never use the same secret key K to two different problems. Thus, for security analysis of the mechanism, we focus on the ciphertext only attack. We do not consider known-plaintext attack in this paper but do allow adversaries to do offline guessing via various problem-dependent information including sizes and signs of the solution, which are not necessarily confidential.

B. Basic Techniques

Before presenting the details of our proposed mechanism, we study in this subsection a few basic techniques and show that the input encryption based on these techniques along may result in an unsatisfactory mechanism. However, the analysis will give insights on how a stronger mechanism should be designed. Note that to simplify the presentation, we assume that the cloud server honestly performs the computation, and defer the discussion on soundness to a later section.

1) Hiding equality constraints $(\mathbf{A}, \mathbf{b})$*:* First of all, a randomly generated $m \times m$ non-singular matrix $\mathbf{Q}$ can be part of the secret key K. The customer can apply the matrix to Eq. (2) for the following constraints transformation,

$$\mathbf{Ax} = \mathbf{b} \quad \Rightarrow \quad \mathbf{A'x} = \mathbf{b'}$$

where $\mathbf{A'} = \mathbf{QA}$ and $\mathbf{b'} = \mathbf{Qb}$.

Since we have assumed that $\mathbf{A}$ has full row rank, $\mathbf{A'}$ must have full row rank. Without knowing $\mathbf{Q}$, it is not possible for one to determine the exact elements of $\mathbf{A}$. However, the nullspaces of $\mathbf{A}$ and $\mathbf{A'}$ remains the same, which may violate the security requirement of some applications. The vector $\mathbf{b}$ is encrypted in a perfect way since it can be mapped to an arbitrary $\mathbf{b'}$ with a proper choice of $\mathbf{Q}$.

2) Hiding inequality constraints $(\mathbf{B})$*:* The customer cannot transform the inequality constraints in the similar way as used for the equality constraints, because for an arbitrary invertible matrix $\mathbf{Q}$, $\mathbf{Bx} \geq \mathbf{0}$ is not equivalent to $\mathbf{QBx} \geq \mathbf{0}$ in general.

To hide $\mathbf{B}$, we can leverage the fact that a feasible solution to Eq. (2) must satisfy the equality constraints. To be more specific, the feasible regions defined by the following two groups of constraints are the same,

$$\begin{cases} \mathbf{Ax} = \mathbf{b} \\ \mathbf{Bx} \geq \mathbf{0} \end{cases} \quad \Rightarrow \quad \begin{cases} \mathbf{Ax} = \mathbf{b} \\ (\mathbf{B} - \boldsymbol{\lambda}\mathbf{A})\mathbf{x} = \mathbf{B'x} \geq \mathbf{0}, \end{cases}$$

where $\boldsymbol{\lambda}$ is a randomly generated $n \times m$ matrix in K satisfying that $|\mathbf{B'}| = |\mathbf{B} - \boldsymbol{\lambda}\mathbf{A}| \neq 0$ and $\boldsymbol{\lambda}\mathbf{b} = \mathbf{0}$. Since the condition $\boldsymbol{\lambda}\mathbf{b} = \mathbf{0}$ is largely underdetermined, it leaves great flexibility to choose $\boldsymbol{\lambda}$ in order to satisfy the above conditions.

3) Hiding objective functions $\mathbf{c}$ *and value* $\mathbf{c}^T\mathbf{x}$*:* Given the widely application of LP, such as the estimation of business annul revenues or personal portfolio holdings etc., the information contained in objective function $\mathbf{c}$ and optimal objective value $\mathbf{c}^T\mathbf{x}$ might be as sensitive as the constraints of $\mathbf{A}, \mathbf{B}, \mathbf{b}$. Thus, they should be protected, too.

To achieve this, we apply constant scaling to the objective function, i.e. a real positive scalar γ is generated randomly as part of encryption key K and $\mathbf{c}$ is replaced by $\gamma\mathbf{c}$. It is not possible to derive the original optimal objective value $\mathbf{c}^T\mathbf{x}$ without knowing γ first, since it can be mapped to any value with the same sign. While hiding the objective value well, this approach does leak structure-wise information of objective function $\mathbf{c}$. Namely, the number and position of zero-elements in $\mathbf{c}$ are not protected. Besides, the ratio between the elements in $\mathbf{c}$ are also preserved after constant scaling.

Summarization of basic techniques Overall, the basic techniques would choose a secret key $K = (\mathbf{Q}, \boldsymbol{\lambda}, \gamma)$ and encrypt the input tuple Φ into $\Phi_K = (\mathbf{A'}, \mathbf{B'}, \mathbf{b'}, \gamma\mathbf{c})$, which gives reasonable strength of problem input hiding. Also, these techniques are clearly correct in the sense that solving Φ_K would give the same optimal solution as solving Φ. However, it also implies that although input privacy is achieved, there is no output privacy. Essentially, it shows that although one can change the constraints to a completely different form, it is not necessary the feasible region defined by the constraints will change, and the adversary can leverage such information to gain knowledge of the original LP problem. Therefore, any secure linear programming mechanism must be able to not only encrypt the constraints but also to encrypt the feasible region defined by the constraints.

C. Enhanced Techniques via Affine Mapping

To enhance the security strength of LP outsourcing, we must be able to change the feasible region of original LP and at the same time hide output vector $\mathbf{x}$ during the problem input encryption. We propose to encrypt the feasible region of Φ by applying an affine mapping on the decision variables $\mathbf{x}$. This design principle is based on the following observation: ideally, if we can arbitrarily transform the feasible area of problem Φ from one vector space to another and keep the mapping function as the secret key, there is no way for cloud server to learn the original feasible area information. Further, such a linear mapping also serves the important purpose of output hiding, as illustrated below.

Let $\mathbf{M}$ be a random $n \times n$ non-singular matrix and $\mathbf{r}$ be an $n \times 1$ vector. The affine mapping defined by $\mathbf{M}$ and $\mathbf{r}$ transforms $\mathbf{x}$ into $\mathbf{y} = \mathbf{M}^{-1}(\mathbf{x} + \mathbf{r})$. Since this mapping is an one-to-one mapping, the LP problem Φ in Eq. (2) can be expressed as the following LP problem of the decision variables $\mathbf{y}$,

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{My} - \mathbf{c}^T\mathbf{r} \\ \text{subject to} \quad & \mathbf{AMy} = \mathbf{b} + \mathbf{Ar} \\ & \mathbf{BMy} \geq \mathbf{Br}. \end{aligned}$$

Next, by using the basic techniques to pick a random non-singular $\mathbf{Q}$ for equality constraints, then $\boldsymbol{\lambda}$ for inequality constraints, and γ for objective function, this LP problem can

be further transformed to,

$$\begin{aligned} \text{minimize} \quad & \gamma \mathbf{c}^T \mathbf{M}\mathbf{y} \\ \text{subject to} \quad & \mathbf{QAMy} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar}), \\ & \mathbf{BMy} - \boldsymbol{\lambda}\mathbf{QAMy} \geq \mathbf{Br} - \boldsymbol{\lambda}\mathbf{Q}(\mathbf{b} + \mathbf{Ar}). \end{aligned}$$

One can denote the constraints of above LP via Eq. (3),

$$\begin{cases} \mathbf{A}' = \mathbf{QAM} \\ \mathbf{B}' = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{QA})\mathbf{M} \\ \mathbf{b}' = \mathbf{Q}(\mathbf{b} + \mathbf{Ar}) \\ \mathbf{c}' = \gamma \mathbf{M}^T \mathbf{c} \end{cases} \tag{3}$$

If the following conditions hold,

$$|\mathbf{B}'| \neq 0, \quad \boldsymbol{\lambda}\mathbf{b}' = \mathbf{Br}, \quad \text{and} \quad \mathbf{b} + \mathbf{Ar} \neq \mathbf{0}, \tag{4}$$

then the LP problem $\Phi_K = (\mathbf{A}', \mathbf{B}', \mathbf{b}', \mathbf{c}')$, which is equivalent to Φ in Eq. (2), can be formulated via Eq. (5),

$$\text{minimize} \quad \mathbf{c'}^T \mathbf{y} \quad \text{subject to} \quad \mathbf{A}'\mathbf{y} = \mathbf{b}', \mathbf{B}'\mathbf{y} \geq \mathbf{0}. \tag{5}$$

Discussion By keeping the randomly selected $\mathbf{M}$ and $\mathbf{r}$ as part of secret key K for affine mapping, it can be ensured that the feasible region of encrypted problem Φ_K no longer contains any resemblance of the feasible area in original problem Φ. As we will show later, both input and output privacy can be achieved by sending Φ_K instead of Φ to the cloud.

D. Result Verification

Till now, we have been assuming the server is honestly performing the computation, while being interested learning information of original LP problem. However, such semi-honest model is not strong enough to capture the adversary behaviors in the real world. In many cases, especially when the computation on the cloud requires a huge amount of computing resources, there exists strong financial incentives for the cloud server to be "lazy". They might either be not willing to commit service-level-agreed computing resources to save cost, or even be malicious just to sabotage any following-up computation at the customers. Since the cloud server promises to solve the LP problem $\Phi_K = (\mathbf{A}', \mathbf{B}', \mathbf{b}', \mathbf{c}')$, we propose to solve the result verification problem by designing a method to verify the correctness of the solution $\mathbf{y}$ of Φ_K. The soundness condition would be a corollary thereafter when we present the whole mechanism in the next section. Note that in our design, the workload required for customers on the result verification is substantially cheaper than solving the LP problem on their own, which ensures the great computation savings for secure LP outsourcing.

The LP problem does not necessarily have an optimal solution. There are three cases as follows: i) *Normal* — there is an optimal solution with finite objective value; ii) *Infeasible* — the constraints cannot be all satisfied at the same time; iii) *Unbounded* — for the standard form in Eq. (1), the objective function can be arbitrarily small while the constraints are all satisfied. Therefore, the result verification method not only needs to verify a solution if the cloud server returns one, but also needs to verify the cases when the cloud server claims that the LP problem is infeasible or unbounded. We will first present the proof Γ that the cloud server should provide and the verification method when the cloud server returns an optimal solution, and then present the proofs and the methods for the other two cases, each of which is built upon the previous one.

1) The normal case: We first assume that the cloud server returns an optimal solution $\mathbf{y}$. In order to verify $\mathbf{y}$ without actually solving the LP problems, we design our method by seeking a set of necessary and sufficient conditions that the optimal solution must satisfy. We derive these conditions from the well-studied duality theory of the LP problems [13]. For the primal LP problem Φ_K defined as Eq. (5), its dual problem is defined as,

$$\text{maximize} \quad \mathbf{b'}^T \mathbf{s} \quad \text{subject to} \quad \mathbf{A'}^T \mathbf{s} + \mathbf{B'}^T \mathbf{t} = \mathbf{c}', \mathbf{t} \geq \mathbf{0}, \tag{6}$$

where $\mathbf{s}$ and $\mathbf{t}$ are the $m \times 1$ and $n \times 1$ vectors of dual decision variables respectively. The strong duality of the LP problems states that if a primal feasible solution $\mathbf{y}$ and a dual feasible solution $(\mathbf{s}, \mathbf{t})$ lead to the same primal and dual objective value, then both $\mathbf{y}$ and $(\mathbf{s}, \mathbf{t})$ are the optimal solutions of the primal and the dual problems respectively [13]. Therefore, we should ask the cloud server to provide the dual optimal solution as part of the proof Γ. Then, the correctness of $\mathbf{y}$ can be verified based on the following conditions,

$$\mathbf{c'}^T \mathbf{y} = \mathbf{b'}^T \mathbf{s}, \mathbf{A}'\mathbf{y} = \mathbf{b}', \mathbf{B}'\mathbf{y} \geq \mathbf{0}, \mathbf{A'}^T \mathbf{s} + \mathbf{B'}^T \mathbf{t} = \mathbf{c}', \mathbf{t} \geq \mathbf{0}. \tag{7}$$

Here, $\mathbf{c'}^T \mathbf{y} = \mathbf{b'}^T \mathbf{s}$ tests the equivalence of primal and dual objective value for strong duality. All the remaining conditions ensure that both $\mathbf{y}$ and $(\mathbf{s}, \mathbf{t})$ are feasible solutions of the primal and dual problems, respectively. Note that due to the possible truncation errors in the computation, the equality test $\mathbf{A}'\mathbf{y} = \mathbf{b}'$ can be achieved in practice by checking whether $||\mathbf{A}'\mathbf{y} - \mathbf{b}'||$ is small enough.

2) The infeasible case: We then assume that the cloud server claims Φ_K to be infeasible. In this case, we leverage the methods to find a feasible solution of a LP problem, usually known as the phase I methods [16]. These methods construct auxiliary LP problems to determine if the original LP problems are feasible or not. We choose the following auxiliary problem,

$$\text{minimize} \quad z \quad \text{subject to} \quad -\mathbf{1}z \leq \mathbf{A}'\mathbf{y} - \mathbf{b}' \leq \mathbf{1}z, \mathbf{B}'\mathbf{y} \geq -\mathbf{1}z. \tag{8}$$

Clearly, this auxiliary LP problem has an optimal solution since it has at least one feasible solution and its objective function is lower-bounded. Further more, one can prove that Eq. (8) has 0 as the optimal objective value if and only if Φ_K is feasible. (See Lemma 29.11 in [17]). Thus, to prove Φ_K is infeasible, the cloud server must prove Eq. (8) has a positive optimal objective value. This can be achieved by including such an optimal solution and a proof of optimality in Γ, which is readily available from the method for the normal case.

3) The unbounded case: Finally, we assume that the cloud server claims Φ_K to be unbounded. The duality theory implies that this case is equivalent to that Φ_K is feasible and the dual problem of Φ_K, i.e. Eq. (6), is infeasible. Therefore, the

cloud server should provide a proof showing that those two conditions hold. It is straight-forward to provide a feasible solution of Φ_K and then to verify it is actually feasible. Based on the method for the infeasible case, the cloud server can prove that Eq. (6) is infeasible by constructing the auxiliary problem of Eq. (6), i.e.,

$$\begin{aligned} \text{minimize} \quad & z \\ \text{subject to} \quad & -\mathbf{1}z \leq \mathbf{A'}^T\mathbf{s} + \mathbf{B'}^T\mathbf{t} - \mathbf{c'} \leq \mathbf{1}z \\ & \mathbf{t} \geq -\mathbf{1}z, \end{aligned} \tag{9}$$

and showing this problem has optimal objective value of 0.

Discussion For all three cases, the cloud server is required to provide correctness proof by proving a normal LP (either Φ_K or some auxiliary LP related to Φ_K) has an optimal solution. Since most common LP algorithms like Simplex and Interior Point methods compute both the primal and dual solutions at the same time [13], providing the dual optimal solution as the optimality proof does not incur any additional overhead for cloud server. Note that the form of auxiliary LP for infeasible/unbounded cases is not unique. In practice, we can adjust it to suit the public solver on cloud, which can be pre-specified by the customer and cloud server with little cost.

E. The Complete Mechanism Description

Based on the previous sections, the proposed mechanism for secure outsourcing of linear programming in the cloud is summarized below.

- KeyGen(1^k): Let $K = (\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma)$. For the system initialization, the customer runs KeyGen(1^k) to randomly generate a secret K, which satisfies Eq. (4).
- ProbEnc(K, Φ): With secret K and original LP problem Φ, the customer runs ProbEnc(K, Φ) to compute the encrypted LP problem $\Phi_K = (\mathbf{A'}, \mathbf{B'}, \mathbf{b'}, \mathbf{c'})$ from Eq. (3).
- ProofGen(Φ_K): The cloud server attempts to solve the LP problem Φ_K in Eq. (5) to obtain the optimal solution $\mathbf{y}$. If the LP problem Φ_K has an optimal solution, Γ should indicate so and include the dual optimal solution $(\mathbf{s}, \mathbf{t})$. If the LP problem Φ_K is infeasible, Γ should indicate so and include the primal and the dual optimal solutions of the auxiliary problem in Eq. (8). If the LP problem Φ_K is unbounded, $\mathbf{y}$ should be a feasible solution of it, and Γ should indicate so and include the primal and the dual optimal solutions of Eq. (9), i.e. the auxiliary problem of the dual problem of Φ_K.
- ResultDec($K, \Phi, \mathbf{y}, \Gamma$): First, the customer verifies $\mathbf{y}$ and Γ according to the various cases. If they are correct, the customer computes $\mathbf{x} = \mathbf{My} - \mathbf{r}$ if there is an optimal solution or reports Φ to be infeasible or unbounded accordingly; otherwise the customer outputs $\perp$, indicating the cloud server was not performing the computation faithfully.

IV. Security Analysis

A. Analysis on Correctness and Soundness Guarantee

We give the analysis on correctness and soundness guarantee via the following two theorems.

Theorem *1: Our scheme is a correct verifiable linear programming outsourcing scheme.*

Proof: The proof consists of two steps. First, we show that for any problem Φ and its encrypted version Φ_K, solution $\mathbf{y}$ computed by honest cloud server will always be verified successfully. This follows directly from the correctness of duality theorem of linear programming. Namely, all conditions derived from duality theorem and auxiliary LP problem construction for result verification are necessary and sufficient.

Next, we show that correctly verified solution $\mathbf{y}$ always corresponds to the optimal solution $\mathbf{x}$ of original problem Φ. For space limit, we only focus on the normal case. The reasoning for infeasible/unbounded cases follows similarly. By way of contradiction, suppose $\mathbf{x} = \mathbf{My} - \mathbf{r}$ is not the optimized solution for Φ. Then, there exists $\mathbf{x}^*$ such that $\mathbf{c}^T\mathbf{x}^* < \mathbf{c}^T\mathbf{x}$, where $\mathbf{Ax}^* = \mathbf{b}$ and $\mathbf{Bx}^* \geq \mathbf{0}$. Since $\mathbf{x}^* = \mathbf{My}^* - \mathbf{r}$, it is straightforward that $\mathbf{c}^T\mathbf{My}^* - \mathbf{c}^T\mathbf{r} = \mathbf{c}^T\mathbf{x}^* < \mathbf{c}^T\mathbf{x} = \mathbf{c}^T\mathbf{My} - \mathbf{c}^T\mathbf{r}$, where $\mathbf{A'y}^* = \mathbf{b'}$ and $\mathbf{B'y}^* \geq \mathbf{0}$. Thus, $\mathbf{y}^*$ is a better solution than $\mathbf{y}$ for problem Φ_K, which contradicts the fact that the optimality of $\mathbf{y}$ has been correctly verified. This completes the proof. ■

Theorem *2:* Our scheme is a sound verifiable linear programming outsourcing scheme.

Proof: Similar to correctness argument, the soundness of the proposed mechanism follows from the facts that the LP problem Φ and Φ_K are equivalent to each other through affine mapping, and all the conditions thereafter for result verification are necessary and sufficient. ■

B. Analysis on Input and Output Privacy Guarantee

We now analyze the input/output privacy guarantee under the aforementioned ciphertext only attack model, due to the one-time-pad type of flexibility of our mechanism (see Section III-A). Specifically, the only information the cloud server obtains is $\Phi_K = (\mathbf{A'}, \mathbf{B'}, \mathbf{b'}, \mathbf{c'})$, and the obvious fact that $\mathbf{A}$ and $\mathbf{B}$ of original LP problem are general full rank matrices as defined in Eq. (2). Note that under such model, offline guessing on problem input/output does not bring cloud server any advantage, since there is no way to justify the validity of the guess. We do not consider known plaintext adversary model and leave the corresponding analysis as future work.

We start from the relationship between the primal problem Φ and its encrypted one Φ_K. First of all, the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ are protected perfectly. Because for $\forall$ $m \times n$ matrix $\mathbf{A'}$ that has the full row rank and $\forall$ $n \times 1$ vector $\mathbf{b'}$, $\exists$ a tuple $(\mathbf{Q}, \mathbf{M}, \mathbf{r})$ that transforms $(\mathbf{A}, \mathbf{b})$ into $(\mathbf{A'}, \mathbf{b'})$. This is straightforward since we can always find invertible matrices $\mathbf{Q}, \mathbf{M}$ for equivalent matrices $\mathbf{A}$ and $\mathbf{A'}$ such that $\mathbf{A'} = \mathbf{QAM}$, and then solve $\mathbf{r}$ from $\mathbf{b'} = \mathbf{Q}(\mathbf{b} + \mathbf{Ar})$. Thus from $(\mathbf{A'}, \mathbf{b'})$, cloud can only derive the rank and size information of original equality constraints $\mathbf{A}$, but nothing else. Secondly, the information of matrix $\mathbf{B}$ is protected by $\mathbf{B'} = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{QA})\mathbf{M}$. Recall that the $n \times m$ matrix $\boldsymbol{\lambda}$ in the condition $\boldsymbol{\lambda}\mathbf{b'} = \mathbf{Br}$ is largely underdetermined. Namely, for

TABLE I: Preliminary Performance Results. Here $t_{original}$, t_{cloud}, and $t_{customer}$ denotes the cloud-side original problem solving time, cloud-side encrypted problem solving time, and customer-side computation time, respectively. The asymmetric speedup captures the customer efficiency gain via LP outsourcing. The cloud efficiency captures the overall computation cost on cloud introduced by solving encrypted LP problem, which should ideally be as close to 1 as possible.

Benchmark		Original Problem	Encrypted Problem		Asymmetric Speedup	Cloud Efficiency
#	size	$t_{original}$ (sec)	t_{cloud} (sec)	$t_{customer}$ (sec)	$\frac{t_{original}}{t_{customer}}$	$\frac{t_{original}}{t_{cloud}}$
1	m = 50, n = 60	0.167	0.170	0.007	26.5 ×	0.981
2	m = 100, n = 120	0.227	0.239	0.005	46.7 ×	0.956
3	m = 200, n = 240	0.630	0.613	0.017	37.3 ×	1.037
4	m = 400, n = 480	3.033	3.671	0.090	33.5 ×	0.835
5	m = 800, n = 960	19.838	23.527	0.569	34.9 ×	0.851
6	m = 1600, n = 1920	171.862	254.012	4.015	42.6 ×	0.690
7	m = 3200, n = 3840	1757.570	2661.360	47.602	36.4 ×	0.745

each $m \times 1$ row vector in $\boldsymbol{\lambda}$, there are $m-1$ elements that can be set freely. Thus, the unlimited choices of $\boldsymbol{\lambda}$, which can be viewed as encryption key with large key space, ensures that $\mathbf{B}$ is well obfuscated. Thirdly, the vector $\mathbf{c}$ is protected well by scaling factor γ and $\mathbf{M}$. By multiplication of matrix $\mathbf{M}$, both the elements and the structure pattern of $\mathbf{c}$ are no longer exposed from $\mathbf{c}' = \gamma \mathbf{M^T c}$. As for the output, since $\mathbf{M}, \mathbf{r}$ is kept as a one-time secret and drawn uniformly at random, deriving $\mathbf{x} = \mathbf{My} - \mathbf{r}$ solely from $\mathbf{y}$ can be hard for cloud.

Given the complementary relationship of primal and dual problem, it is also worth looking into the input/output privacy guarantee from dual problems of both Φ and Φ_K. Same as Eq. (6), the dual problem of Φ is defined as,

$$\text{maximize} \quad \mathbf{b}^T\boldsymbol{\alpha} \quad \text{subject to} \quad \mathbf{A}^T\boldsymbol{\alpha}+\mathbf{B}^T\boldsymbol{\beta} = \mathbf{c}, \boldsymbol{\beta} \geq \mathbf{0}, \quad (10)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the $m \times 1$ and $n \times 1$ vectors of dual decision variables respectively. Clearly, the analysis for primal problem Φ's input privacy guarantee still holds for its dual problem input $(\mathbf{A}, \mathbf{B}, \mathbf{b}, \mathbf{c})$. As for the output privacy, we plug Eq. (3) into Φ_K's dual problem defined in Eq. (6) and rearrange it as,

$$\begin{aligned} \text{maximize} \quad & [\mathbf{Q}(\mathbf{b} + \mathbf{Ar})]^T\mathbf{s} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{Q}^T(\mathbf{s} - \boldsymbol{\lambda}^T\mathbf{t}) + \mathbf{B}^T\mathbf{t} = \gamma\mathbf{c} \\ & \mathbf{t} \geq \mathbf{0}, \end{aligned} \quad (11)$$

Note that $\mathbf{M}^T$ in the equality constraint is canceled out during the rearrangement. Comparing Eq. (10) and Eq. (11), we derive the linear mapping between $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $(\mathbf{s}, \mathbf{t})$ as,

$$\boldsymbol{\alpha} = \frac{1}{\gamma}\mathbf{Q}^T(\mathbf{s} - \boldsymbol{\lambda}^T\mathbf{t}), \ \boldsymbol{\beta} = \frac{1}{\gamma}\mathbf{t} \quad (12)$$

Following similar reasoning for Φ's output privacy and analysis for hiding objective function $\mathbf{c}$ in basic techniques (Section III-B3), the dual decision variables $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of original problem Φ is protected well by the random choice of $(\mathbf{Q}, \boldsymbol{\lambda}, \gamma)$.

V. Performance Analysis

A. Theoretic Analysis

1) Customer Side Overhead: For the three customer side algorithms KeyGen, ProbEnc, and ResultDec, it is straightforward that the most time-consuming operations are the matrix-matrix multiplications in problem encryption algorithm ProbEnc. Since $m \leq n$, the time complexity for the customer local computation is thus asymptotically the same as matrix-matrix multiplication, i.e., $O(n^\rho)$ for some $2 < \rho \leq 3$. In our experiment, the matrix multiplication is implemented via standard cubic-time method, thus the overall computation overhead is $O(n^3)$. However, other more efficient matrix multiplication algorithms can also be adopted, such as the Strassen's algorithm with time complexity $O(n^{2.81})$ [18] or the Coppersmith-Winograd algorithm [19] in $O(n^{2.376})$. In either case, the customer side efficiency can be further improved.

2) Server Side Overhead: For cloud server, its only computation overhead is to solve the encrypted LP problem Φ_K as well as generating the result proof Γ, both of which correspond to the algorithm ProofGen. If the encrypted LP problem Φ_K belongs to normal case, cloud server just solves it with the dual optimal solution as the result proof Γ, which is usually readily available in the current LP solving algorithms and incurs no additional cost for cloud (see Section III-D). If the encrypted problem Φ_K does not have an optimal solution, additional auxiliary LP problems can be solved to provide a proof. Because for general LP solvers, phase I method (solving the auxiliary LP) is always executed at first to determine the initial feasible solution [16], proving the auxiliary LP with optimal solutions also introduces little additional overhead. Thus, in all the cases, the computation complexity of the cloud server is asymptotically the same as to solve a normal LP problem, which usually requires more than $O(n^3)$ time [13].

Obviously, the customer will not spend more time to encrypt the problem and solve the problem in the cloud than to solve the problem on his own. Therefore, in theory, the proposed mechanism would allow the customer to outsource their LP problems to the cloud and gain great computation savings.

B. Experiment Results

We now assess the practical efficiency of the proposed secure and verifiable LP outsourcing scheme with experiments. We implement the proposed mechanism including both the customer and the cloud side processes in Matlab and utilize the MOSEK optimization [20] through its Matlab interface to

solve the original LP problem Φ and encrypted LP problem Φ_K. Both customer and cloud server computations in our experiment are conducted on the same workstation with an Intel Core 2 Duo processor running at 1.86 GHz with 4 GB RAM. In this way, the practical efficiency of the proposed mechanism can be assessed without a real cloud environment. We also ignore the communication latency between the customers and the cloud for this application since the computation dominates the running time as evidenced by our experiments.

Our randomly generated test benchmark covers the small and medium sized problems, where m and n are increased from 50 to 3200 and 60 to 3840, respectively. All these benchmarks are for the normal cases with feasible optimal solutions. Since in practice the infeasible/unbounded cases for LP computations are very rare, we do not conduct those experiments for the current preliminary work and leave it as one of our future tasks. Table I gives our experimental results, where each entry in the table represents the mean of 20 trials.

In this table, the sizes of the original LP problems are reported in the first two columns. The times to solve the original LP problem in seconds, $t_{original}$, are reported in the third column. The times to solve the encrypted LP problem in seconds are reported in the fourth and fifth columns, separated into the time for the cloud server t_{cloud} and the time for the customer $t_{customer}$. Note that since each KeyGen would generate a different key, the encrypted LP problem Φ_K generated by ProbEnc would be different and thus result in a different running time to solve it. The t_{cloud} and $t_{customer}$ reported in Table I are thus the average of multiple trials. We propose to assess the practical efficiency by two characteristics calculated from $t_{original}$, t_{cloud}, and $t_{customer}$. The *Asymmetric Speedup*, calculated as $\frac{t_{original}}{t_{customer}}$, represents the savings of the computing resources for the customers to outsource the LP problems to the cloud using the proposed mechanism. The *Cloud Efficiency*, calculated as $\frac{t_{original}}{t_{cloud}}$, represents the overhead introduced to the overall computation by the proposed mechanism. It can be seen from the table that we can always achieve more than $30\times$ savings when the sizes of the original LP problems are not too small. On the other hand, from the last column, we can claim that for the whole system including the customers and the cloud, the proposed mechanism will not introduce a substantial amount of overhead. It thus confirms that secure outsourcing LP in cloud computing is economically viable.

VI. Related Work

A. Work on Secure Computation Outsourcing

General secure computation outsourcing that fulfills all aforementioned requirements, such as input/output privacy and correctness/soundness guarantee has been shown feasible in theory by Gennaro et al. [9]. However, it is currently not practical due to its huge computation complexity. Instead of outsourcing general functions, in the security community, Atallah et al. explore a list of work [5], [7], [8], [10] for securely outsourcing specific applications. The customized solutions are expected to be more efficient than the general way of constructing the circuits. In [5], they give the first investigation of secure outsourcing of numerical and scientific computation. A set of problem dependent disguising techniques are proposed for different scientific applications like linear algebra, sorting, string pattern matching, etc. However, these disguise techniques explicitly allow information disclosure to certain degree. Besides, they do not handle the important case of result verification, which in our work is bundled into the design and comes at close-to-zero additional cost. Later on in [7] and [8], Atallah et al. give two protocol designs for both secure sequence comparison outsourcing and secure algebraic computation outsourcing. However, both protocols use heavy cryptographic primitive such as homomorphic encryptions [21] and/or oblivious transfer [22] and do not scale well for large problem set. In addition, both designs are built upon the assumption of two non-colluding servers and thus vulnerable to colluding attacks. Based on the same assumption, Hohenberger et al. [6] provide protocols for secure outsourcing of modular exponentiation, which is considered as prohibitively expensive in most public-key cryptography operations. Very recently, Atallah [10] et al. give a provably secure protocol for secure outsourcing matrix multiplications based on secret sharing [23]. While this work outperforms their previous work [8] in the sense of single server assumption and computation efficiency (no expensive cryptographic primitives), the drawback is the large communication overhead. Namely, due to secret sharing technique, all scalar operations in original matrix multiplication are expanded to polynomials, introducing significant amount of overhead. Considering the case of the result verification, the communication overhead must be further doubled, due to the introducing of additional pre-computed "random noise" matrices.

In short, these solutions, although elegant, are still not efficient enough for immediate practical uses, which we aim to address for the secure LP outsourcing in this paper.

B. Work on Secure Multiparty Computation

Another large existing list of work that relates to (but is also significantly different from) ours is Secure Multi-party Computation (SMC), first introduced by Yao [11]. SMC allows two or more parties to jointly compute some general function while hiding their inputs to each other. As general SMC can be very inefficient, Du and Atallah et. al. have proposed a series of customized solutions under the SMC context to a spectrum of special computation problems, such as privacy-preserving cooperative statistical analysis, scientific computation, geometric computations, sequence comparisons, etc. [24]. However, directly applying these approaches to the cloud computing model for secure computation outsourcing would still be problematic. The major reason is that they did not address the asymmetry among the computational powers possessed by cloud and the customers, i.e., all these schemes in the context of SMC impose each involved parties comparable computation burdens, which we specifically avoid in the mechanism design by shifting as much as possible computation burden to cloud only. Another reason is the asymmetric security requirement.

In SMC no single involved party knows all the problem input information, making result verification a very difficult task. But in our model, we can explicitly exploit the fact that the customer knows all input information and thus design efficient result verification mechanism.

Under the SMC model, Li and Atallah [25] give the first study for secure and collaborative computation of linear programming. Their solution is based on the additive split of the constraint matrix between two involved parties, followed by a series of interactive (and arguably heavy) cryptographic protocols collaboratively executed in each iteration step of the Simplex Algorithm. The work does not provide practical performance for big size problems. Besides, they only consider honest-but-curious model and thus do not guarantee that the final solution is optimal. Following the same framework, Toft [26] also proposes a secure Simplex Algorithm based on secret sharing, which outperforms the one in [25] with slightly better protocol complexity. In [27], Vaidya presented a revised Simplex Algorithm based on secure scalar product and secure comparison protocols. Very recently, Catrina et al. [28] propose a secure multiparty linear programming via fixed-point arithmetics. Though more efficient than the work in [25], [26], their approach still only works for relatively small size problems, and does not enjoy the computational simplicity as our mechanism. Moreover, all these designs have the computation asymmetry issue mentioned previously, and thus are not suitable for the computation outsourcing scenario.

In other related work for secure linear programming, Bednarz et al. [29] propose to use generalized permutation matrices with positive elements to disguise the linear constraints. But such permutation matrices explicitly preserve the number of zero elements (aka. sparsity) of the original constraint matrix, and thus the input protection is not complete.

VII. Concluding Remarks

In this paper, for the first time, we formalized the problem of securely outsourcing LP computations in cloud computing, and provided such a practical mechanism design which fulfills input/output privacy, cheating resilience, and efficiency. By explicitly decomposing LP computation outsourcing into public LP solvers and private data, our mechanism design is able to explore appropriate security/efficiency tradeoffs via higher level LP computation than the general circuit representation. We developed problem transformation techniques that enable customers to secretly transform the original LP into some arbitrary one while protecting sensitive input/output information. We also investigated duality theorem and derived a set of necessary and sufficient condition for result verification. Such a cheating resilience design can be bundled in the overall mechanism with close-to-zero additional overhead. Both security analysis and experiment results demonstrates the immediate practicality of the proposed mechanism.

We plan to investigate some interesting future work as follows: 1) establish formal security framework; 2) devise robust algorithms to achieve numerical stability; 3) extend our result to non-linear programming outsourcing in cloud.

Acknowledgement

This work was supported in part by the US National Science Foundation under grant CNS-0831963.

References

[1] P. Mell and T. Grance, "Draft nist working definition of cloud computing," Referenced on Jan. 23rd, 2010 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2010.

[2] Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing," 2009, online at http://www.cloudsecurityalliance.org.

[3] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[4] Sun Microsystems, Inc., "Building customer trust in cloud computing with transparent security," 2009, online at https://www.sun.com/offers/details/sun_transparency.xml.

[5] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, "Secure outsourcing of scientific computations," *Advances in Computers*, vol. 54, pp. 216–272, 2001.

[6] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. of TCC*, 2005, pp. 264–282.

[7] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int'l J. Inf. Sec.*, vol. 4, no. 4, pp. 277–287, 2005.

[8] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. of Int'l Conf. on Privacy, Security, and Trust (PST)*, 2008, pp. 240–245.

[9] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. of CRYPTO*, Aug. 2010.

[10] M. Atallah and K. Frikken, "Securely outsourcing linear algebra computations," in *Proc. of ASIACCS*, 2010, pp. 48–59.

[11] A. C.-C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. of FOCS*, 1982, pp. 160–164.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc of STOC*, 2009, pp. 169–178.

[13] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. Springer, 2008.

[14] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. of ICDCS*, 2010.

[15] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, March 2010.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT press, 2008.

[18] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, pp. 354–356, 1969.

[19] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. of STOC*, 1987, pp. 1–6.

[20] MOSEK ApS, "The MOSEK Optimization Software," Online at http://www.mosek.com/, 2010.

[21] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. of EUROCRYPT*, 1999, pp. 223–238.

[22] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[23] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[24] W. Du and M. J. Atallah, "Secure multi-party computation problems and their applications: a review and open problems," in *Proc. of New Security Paradigms Workshop (NSPW)*, 2001, pp. 13–22.

[25] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Proc. of Int'l Conf. on Collaborative Computing*, 2006.

[26] T. Toft, "Solving linear programs using multiparty computation," in *Proc. of Financial Cryptography and Data Security*, 2009, pp. 90–107.

[27] J. Vaidya, "A secure revised simplex algorithm for privacy-preserving linear programming," in *Proc. of IEEE Conf. on Advanced Information Networking and Applications (AINA)*, 2009.

[28] O. Catrina and S. De Hoogh, "Secure multiparty linear programming using fixed-point arithmetic," in *Proc. of ESORICS*, 2010, pp. 134–150.

[29] A. Bednarz, N. Bean, and M. Roughan, "Hiccups on the road to privacy-preserving linear programming," in *Proc. of ACM workshop on Privacy in the Electronic Society (WPES)*, 2009.