Secure Optimization Computation Outsourcing in Cloud Computing: A Case Study of Linear Programming

Cong Wang, *Member, IEEE,* Kui Ren, *Senior Member, IEEE,* and Jia Wang, *Member, IEEE*

Abstract—Cloud computing enables an economically promising paradigm of computation outsourcing. However, how to protect customers confidential data processed and generated during the computation is becoming the major security concern. Focusing on engineering computing and optimization tasks, this paper investigates secure outsourcing of widely applicable linear programming (LP) computations. Our mechanism design explicitly decomposes 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 computation than the general circuit representation. Specifically, by formulating private LP problem as a set of matrices/vectors, we develop efficient privacy-preserving problem transformation techniques, which allow customers to transform the original LP into some random one while protecting sensitive input/output information. To validate the computation result, we further explore the fundamental duality theorem of LP and derive the necessary and sufficient conditions that correct results must satisfy. Such result verification mechanism is very 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.

Index Terms—Confidential data, computation outsourcing, optimization, cloud computing, linear programming

1 Introduction

Cloud provides robust computing power to the society with reduced cost [2]. One fundamental advantage enabled by cloud is computation outsourcing. With outsourcing, customers are no longer limited by their computationally weak devices but can enjoy the abundant computing resources from cloud in an economically pay-per-use manner. Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customers' direct control over the systems that process and generate their data during the computation, which inevitably brings in new security concerns and challenges towards this promising computing model [3]. 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 [3] 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 [4], 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 [3]. 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 ho-

- *Cong Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: congwang@cityu.edu.hk.*
- *Kui Ren is with the Department of Computer Science and Engineering, The State University of New York at Buffalo. E-mail: kuiren@buffalo.edu.*
- *Jia Wang is with the Department of Electrical and Computer Engineering, Illinois Institute of Technology. E-mail: jwang@ece.iit.edu.*

A preliminary version [1] of this paper was presented at the 30th IEEE Conference on Computer Communications (INFOCOM'11).

momorphic 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 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 7). 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/models, such as packet routing, flow control, power management of data centers, etc. [13]. Because LP computations may require a substantial amount of computational power and usually involve confidential data, say local customer's business annul revenues, personal portfolio holdings, finance investment strategies, etc., depending on the different application contexts, 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 random 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. We summarize our contributions as follows:

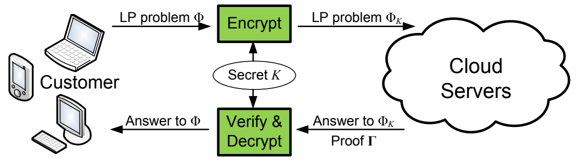


Fig. 1: Architecture of secure outsourcing LP in Cloud

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 demonstrates the immediate practicality: our mechanism can always help customers achieve more than $50\times$ savings when the sizes of the original LP problems (with feasible solutions) are not too small, while introducing no substantial overhead on the cloud.

The paper organizations: Section 2 introduces the system and threat model, and our design goals. Then we provide the detailed mechanism description in Section 3, followed by the security analysis in Section 4 and some further considerations in Section 5. We give performance evaluation in Section 6, followed by Section 7 on related work. Finally, Section 8 concludes the whole paper.

2 Problem Statement

2.1 System and Threat Model

Our computation outsourcing architecture is illustrated in Fig. 1: 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 the *cloud server* (CS) for solving the LP computation and leverages its robust 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 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 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 received and the encrypted output produced to learn sensitive information as in the semi-honest model. In addition, CS can also behave unfaithfully to 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.

2.2 Design Goals

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

Correctness: Any cloud server that faithfully follows the mechanism must produce an output that can be decrypted and verified successfully by the customer.

Soundness: No cloud server can generate an incorrect output that can be decrypted and verified successfully by the customer with non-negligible probability.

Input/output privacy: No sensitive information from the customer's private data can be derived by the cloud server during performing the LP computation.

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.

2.3 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,

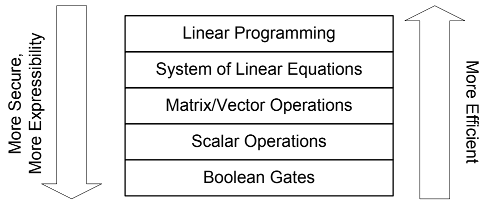


Fig. 2: A hierarchy of computations and mechanisms

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

Here $\mathbf{x}$ is an $n \times 1$ vector of decision variables, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$ is an $n \times 1$ column vector, and $\mathbf{b}$ is an $m \times 1$ column vector. 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 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 solution $\mathbf{x}$ as output.

3 THE PROPOSED SCHEMES

This section presents our *complete LP outsourcing* solution for – not only the privacy protection of problem input/output, but also its efficient result checking. We first give an overview of our methodology, and systematically justify how we can leverage the security/efficiency tradeoffs through properly-chosen problem decomposition. We next present the design framework for secure LP outsourcing and discuss a few basic techniques and their demerits. This leads to a stronger problem transformation design utilizing affine mapping. We then discuss effective result verification by leveraging the duality of LP. Finally, we give the full scheme description.

3.1 Methodology Overview

Secure LP outsourcing in cloud can be represented by decomposing LP computation into public LP solvers running on the cloud and private data owned by the customer. Because different decompositions of LP usually lead to different trade-offs among efficiency and security guarantees, how to choose the right one that is most suitable for our design goal is thus of critical importance. To systematically study the difference, we organize the different decompositions into a hierarchy, as shown in Fig. 2, which ensembles the usual way that a computation is specified: a computation at a higher abstraction level is made up from the computations at lower abstraction levels. At higher abstraction levels, more information about the computations becomes public so that security guarantees become weaker. But more structures become available, and the mechanisms become more efficient. At lower abstraction levels, the structures become generic, but less information is available to the cloud so that stronger security guarantees could be achieved at the cost of efficiency.

Because we aim to design practically efficient mechanisms of secure LP outsourcing, we focus on the top level of the hierarchy in Fig. 2. Namely, we will study problem transformation techniques that enable customers to secretly transform the original LP into some random one to achieve the secure LP outsourcing design.

3.2 Mechanism Design Framework

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 instantiated later.

- KeyGen(1^κ) $\rightarrow \{K\}$. *This is a randomized key generation algorithm which takes a system security parameter κ, and returns a secret key K that is used later by customer to encrypt the target LP problem.*
- ProbEnc(K, Φ) $\rightarrow \{\Phi_K\}$. *This algorithm encrypts the input tuple Φ into Φ_K with the secret key K. According to problem transformation, the encrypted output Φ_K has the same form as Φ, and thus defines the problem to be solved in the cloud.*
- ProofGen(Φ_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 shall never use the same secret key K for two different problems. Thus, for security analysis of the mechanism, we focus on the ciphertext only attack, where the adversary is assumed to only have access to the encrypted problem. We do not consider known-plaintext attack in this paper, where the attacker has access to both the original problem and its encrypted version, 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.

3.3 Basic Techniques

We first 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. To simplify the presentation, we assume a semi-honest cloud here, and defer the soundness discussion to a later section.

3.3.1 *Hiding equality constraints* $(\mathbf{A}, \mathbf{b})$

Firstly, 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 transformation, $\mathbf{Ax} = \mathbf{b} \ \Rightarrow \ \mathbf{A}'\mathbf{x} = \mathbf{b}'$, where $\mathbf{A}' = \mathbf{QA}$ and $\mathbf{b}' = \mathbf{Qb}$.

As we assume $\mathbf{A}$ 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 $\mathbf{Q}$.

3.3.2 *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,

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

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 $\boldsymbol{\lambda}\mathbf{b} = \mathbf{0}$ is largely underdetermined, it leaves great flexibility to choose $\boldsymbol{\lambda}$ so as to satisfy the above conditions.

3.3.3 *Hiding objective functions* $\mathbf{c}$ *and value* $\mathbf{c}^T\mathbf{x}$

Given the wide application of LP, such as the estimation of business revenues or personal portfolio holdings, the information in objective function $\mathbf{c}$ and optimal objective value $\mathbf{c}^T\mathbf{x}$ might be sensitive and need protection, 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.

3.4 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 $\mathbf{y}$,

$$\begin{aligned} &\text{minimize} && \mathbf{c}^T\mathbf{M}\mathbf{y} - \mathbf{c}^T\mathbf{r} \\ &\text{subject to} && \mathbf{A}\mathbf{M}\mathbf{y} = \mathbf{b} + \mathbf{A}\mathbf{r},\ \mathbf{B}\mathbf{M}\mathbf{y} \geq \mathbf{B}\mathbf{r}. \end{aligned}$$

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,

$$\begin{aligned} &\text{minimize} && \gamma\mathbf{c}^T\mathbf{M}\mathbf{y} \\ &\text{subject to} && \mathbf{Q}\mathbf{A}\mathbf{M}\mathbf{y} = \mathbf{Q}(\mathbf{b} + \mathbf{A}\mathbf{r}), \\ & && \mathbf{B}\mathbf{M}\mathbf{y} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A}\mathbf{M}\mathbf{y} \geq \mathbf{B}\mathbf{r} - \boldsymbol{\lambda}\mathbf{Q}(\mathbf{b} + \mathbf{A}\mathbf{r}). \end{aligned}$$

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

$$\mathbf{A}' = \mathbf{Q}\mathbf{A}\mathbf{M},\ \mathbf{B}' = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A})\mathbf{M}, \quad \mathbf{b}' = \mathbf{Q}(\mathbf{b} + \mathbf{A}\mathbf{r}), \mathbf{c}' = \gamma\mathbf{M}^T\mathbf{c} \quad (3)$$

If the following conditions hold,

$$|\mathbf{B}'| \neq 0, \quad \boldsymbol{\lambda}\mathbf{b}' = \mathbf{B}\mathbf{r}, \quad \text{and} \quad \mathbf{b} + \mathbf{A}\mathbf{r} \neq \mathbf{0}, \qquad (4)$$

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),

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

Remark. Note that during the transformation, we ignore the constant term $\mathbf{c}^T\mathbf{r}$ as it has nothing to do with final solution $\mathbf{y}$, and we can always add it back to get the original objective value. 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.

3.5 Result Verification

Till now, we have been assuming the server is honestly performing the computation. However, such a model is not strong enough to capture the adversary behaviors in the real world. In many cases, especially when the computation requires a huge amount of computing resources, there exists strong financial incentives for the cloud to be "lazy". It 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.

3.5.1 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,

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

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 $\mathbf{A'y} = \mathbf{b'}$ can be achieved by checking whether $||\mathbf{A'y} - \mathbf{b'}||$ is small enough.

3.5.2 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,

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

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.

3.5.3 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.,

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

and showing it has a positive optimal objective value.

Remark. 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 with little cost.

3.6 The Complete Mechanism Description

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

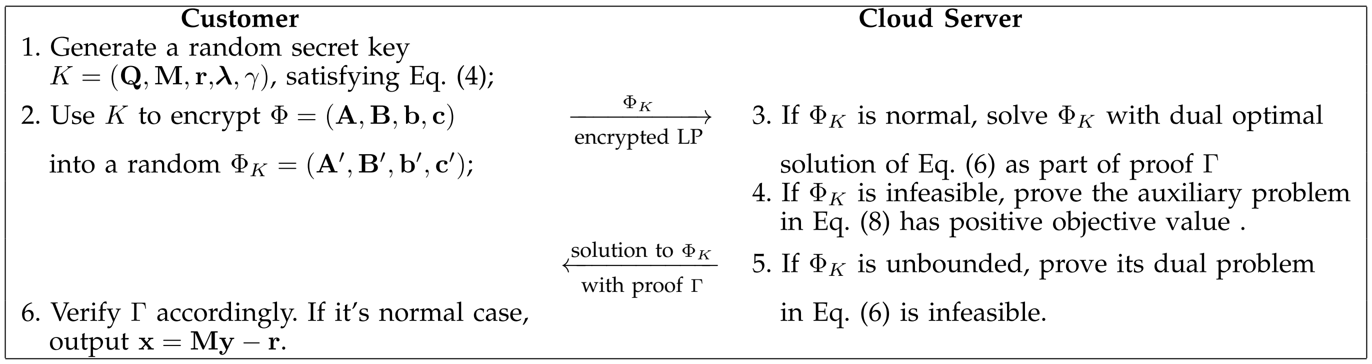
- KeyGen(1^κ): Let $K = (\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma)$. For the system initialization, the customer runs KeyGen(1^κ) to randomly generate a secret K, which satisfies Eq. (4).
- ProbEnc(K, Φ): With secret K and original LP problem Φ, the customer runs ProbEnc(K, Φ) to compute the encrypted LP $\Phi_K = (\mathbf{A'}, \mathbf{B'}, \mathbf{b'}, \mathbf{c'})$ 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.

4 Security Analysis

4.1 Analysis on Correctness and Soundness

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.*



Customer

1. Generate a random secret key $K = (\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma)$, satisfying Eq. (4);
2. Use K to encrypt $\Phi = (\mathbf{A}, \mathbf{B}, \mathbf{b}, \mathbf{c})$ into a random $\Phi_K = (\mathbf{A}', \mathbf{B}', \mathbf{b}', \mathbf{c}')$;

$\xrightarrow[\text{encrypted LP}]{\Phi_K}$

Cloud Server

3. If Φ_K is normal, solve Φ_K with dual optimal solution of Eq. (6) as part of proof Γ
4. If Φ_K is infeasible, prove the auxiliary problem in Eq. (8) has positive objective value .
5. If Φ_K is unbounded, prove its dual problem in Eq. (6) is infeasible.

$\xleftarrow[\text{with proof } \Gamma]{\text{solution to } \Phi_K}$

6. Verify Γ accordingly. If it's normal case, output $\mathbf{x} = \mathbf{M}\mathbf{y} - \mathbf{r}$.

Fig. 3: The complete mechanism for secure LP outsourcing in cloud

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 $\mathbf{x} = \mathbf{M}\mathbf{y} - \mathbf{r}$ 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 $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ and $\mathbf{B}\mathbf{x}^* \geq \mathbf{0}$. Since $\mathbf{x}^* = \mathbf{M}\mathbf{y}^* - \mathbf{r}$, it is straightforward that $\mathbf{c}^T\mathbf{M}\mathbf{y}^* - \mathbf{c}^T\mathbf{r} = \mathbf{c}^T\mathbf{x}^* < \mathbf{c}^T\mathbf{x} = \mathbf{c}^T\mathbf{M}\mathbf{y} - \mathbf{c}^T\mathbf{r}$, where $\mathbf{A}'\mathbf{y}^* = \mathbf{b}'$ and $\mathbf{B}'\mathbf{y}^* \geq \mathbf{0}$. 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: 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. □

4.2 Analysis on Input and Output Privacy

We now analyze the input/output privacy guarantee under the aforementioned ciphertext only attack model. 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 in our model no secret transformation key shall be used twice (see Section 3.2). 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 assume our system uses finite precision floating numbers, and each entry x_i of the original solution $\mathbf{x}$ should be in range $(-L, L)$, where $L = \mathsf{poly}(\kappa)$ with κ as our security parameter and poly as a polynomial function. Let the system input $m \leq n = \theta(\kappa)$. Below we show that neither the original problem output/input nor the secret key can be deduced by adversary with non-negligible probability.

First, we argue that the transformed optimal solution $\mathbf{y}$ does not reveal $\mathbf{x}$. Recall that $\mathbf{x} = \mathbf{M}\mathbf{y} - \mathbf{r}$ and $\mathbf{y} = \mathbf{M^{-1}}(\mathbf{x} + \mathbf{r})$, where $\mathbf{M}$ is a randomly chosen invertible matrix. As for the random $\mathbf{r}$, we can uniformly pick each entry r_i of $\mathbf{r}$ from a relatively big interval $[-2^\kappa, 2^\kappa]$ with fixed precision. Denote uniform distribution from $[-2^\kappa, 2^\kappa]$ with fixed precision as $\mathcal{U}(-2^\kappa, 2^\kappa)$. If we pick a random vector $\boldsymbol{\eta}$, where each entry in $\boldsymbol{\eta}$ is sampled from the uniform distribution $\mathcal{U}(-2^\kappa, 2^\kappa)$, we show that the distribution of $\mathbf{x} + \mathbf{r}$ is statistically close to $\boldsymbol{\eta}$.

Theorem 3: The statistical distance

$$\mathsf{SD}(\mathbf{x} + \mathbf{r}, \boldsymbol{\eta}) \leq \mu(\kappa) \ ,$$

where $\mu(\kappa)$ is a negligible function.

Proof: We first show that for each individual entry, $\mathsf{SD}(x_i + r_i, \eta_i)$, $i \in \{1, \ldots, n\}$, is negligible. We have two hypothesises $\mathcal{H}_0$ and $\mathcal{H}_1$, whose ranges are $[-2^\kappa - L, 2^\kappa + L]$ and $[-2^\kappa, 2^\kappa]$, respectively. It is easy to see the optimal distinguishing strategy is to output 0 if the input is from $[-2^\kappa - L, -2^\kappa)$ and $(2^\kappa, 2^\kappa + L]$, and output a random guess $b \leftarrow \{0, 1\}$, otherwise. This distinguisher's success probability is

$$\begin{aligned} p &= \frac{1}{2} + \Pr[y_i + r_i \in [-2^\kappa - L, -2^\kappa)] \\ &\quad + \Pr[y_i + r_i \in (2^\kappa, 2^\kappa + L]] \\ &\leq \frac{1}{2} + \frac{2L}{2^\kappa} = \frac{1}{2} + \mu'(\kappa), \end{aligned}$$

where μ' is a negligible function. Then by applying union bound, we have $\mathsf{SD}(\mathbf{x} + \mathbf{r}, \boldsymbol{\eta}) \leq \mu(\kappa)$ where $\mu(\kappa) = n * \mu'(\kappa)$ as claimed. □

Therefore, from the cloud's view, $\mathbf{x} + \mathbf{r}$ and $\boldsymbol{\eta}$ are statistically indistinguishable. Equivalently, the cloud's view of $\mathbf{y} = \mathbf{M^{-1}}(\mathbf{x} + \mathbf{r})$ and $\mathbf{y}^* = \mathbf{M^{-1}}\boldsymbol{\eta}$ is statistically indistinguishable. Thus $\mathbf{y}$ does not reveal $\mathbf{x}$. Similarly, $\mathbf{b}' = \mathbf{Q}(\mathbf{b} + \mathbf{A}\mathbf{r})$ statistically hides $\mathbf{b}$. Since $\mathbf{b} = \mathbf{A}\mathbf{x}$, we have $\mathbf{b}' = \mathbf{Q}\mathbf{A}(\mathbf{x} + \mathbf{r})$. Again, since each entry of $\mathbf{r}$ is sampled from the uniform distribution $\mathcal{U}(-2^\kappa, 2^\kappa)$, by Theorem 3, we can replace $\mathbf{x} + \mathbf{r}$ with $\mathbf{r}$. The cloud's views of $\mathbf{b}' = \mathbf{Q}\mathbf{A}(\mathbf{x} + \mathbf{r})$ and $\mathbf{b}^* = \mathbf{Q}\mathbf{A}\mathbf{r}$ are statistically indistinguishable. Hence, $\mathbf{b}'$ does not reveal $\mathbf{b}$.

Next we show the individual protection of $\mathbf{A}, \mathbf{B}$, and $\mathbf{c}$. For $\mathbf{A}' = \mathbf{QAM}$, since both $\mathbf{Q}$ and $\mathbf{M}$ are randomly chosen, we can map $\mathbf{A}$ to any given $\mathbf{A}'$ with the same size and rank with $\mathbf{A}$. Thus, from $\mathbf{A}'$ cloud can only derive the rank and size information of original $\mathbf{A}$ but nothing else. For $\mathbf{B}' = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{QA})\mathbf{M}$, we have the $n \times m$ matrix $\boldsymbol{\lambda}$ in the condition $\boldsymbol{\lambda}\mathbf{b}' = \mathbf{Br}$ is largely underdetermined. Namely, for 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. For $\mathbf{c}' = \gamma\mathbf{M}^\mathbf{T}\mathbf{c}$, 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.

Finally, we show that given $\mathbf{A}', \mathbf{B}', \mathbf{b}', \mathbf{c}'$, it is infeasible to solve the key components $(\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma)$. Indeed, it is well known that solving the system of non-linear equation is NP hard [18], [19], and the problem size is $O(n) = \theta(\kappa)$. Thus solving this underdetermined system of non-linear equation takes at least $\mathsf{exp}(\kappa)$ running time. Hence, polynomial running time adversary has negligible chance to succeed.

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,

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

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,

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

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,

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

Following similar reasoning for Φ's output privacy and analysis for hiding objective function $\mathbf{c}$ in basic techniques (Section 3.3.3), the dual decision variables $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of original problem Φ is protected well by the random choice of $(\mathbf{Q}, \boldsymbol{\lambda}, \gamma)$.

Remark. Given the recent advancement of FHE and its applications in secure outsourcing designs, e.g., [9], below we present a comprehensive comparison of our security design and FHE related solutions.

1) FHE is essentially a generic encryption that supports evaluating arbitrary functions over the encrypted data. Our random transformation, on the other hand, is a customized security design that specifically focuses on LP. It combines a suite of domain specific transformation techniques to achieve the protection.

2) As an encryption design, the ciphertext from FHE cannot be broken without its encryption key, even if the same encryption key has been used multiple times. In our customized design, we must not reuse the secret transformation key, as that might give further advantage to the adversary to reverse-engineer our random transformation and hence expose the original LP.

3) Our design and FHE have different adversary models. FHE can be proved semantically secure against chosen plaintext attacks (CPA) (but not chosen ciphertext attacks) [4], [12]. However, due to our domain specific design and its requirement of one-time key, we just show our security analysis against ciphertext only attack, which is a weaker model compared to FHE. Because in our applications customers themselves generate and encrypt LP problems before outsourcing, our security model is expected to work well in practice. Our performance is also expected to be much faster.

4) The levels of security guarantee to be achieved can be different. For FHE related solutions, it is possible to combine FHE with encrypted garbled circuits [11] to achieve a secure outsourcing design where the cloud learns nothing about the circuit or the input/output protected by FHE. This is arguably the most complete security protection for any secure outsourcing solutions. Note that in our transformation designs, we explicitly reveal that the underlying computation is LP. But this tradeoff is expected to bring us huge efficiency gain in the security design as shown soon.

5 Further Investigations

From above discussions, we know that both input and output of the problem Φ has been protected well via the random choice of keys $(\mathbf{Q}, \mathbf{M}, \mathbf{r}, \boldsymbol{\lambda}, \gamma)$. However, it is not yet clear what the underlying connection between the two LP problems Φ and Φ_K is and how that relationship could benefit our mechanism design. In this section, we thoroughly investigate this problem through rigorous mathematical reasoning. In addition, we discuss how the uncovered results could affect the possible information leakage on some special cases, and how we can effectively address them via lightweight techniques.

5.1 Connections between Φ and Φ_K

To better understand the connection between Φ and Φ_K, we start by introducing two new problems Ψ and Ψ_K, which are derived from Φ and Φ_K respectively by using one more step of transformation. Specifically, for Φ defined in Eq. (2), we define another LP problem Ψ via the transformation $\mathbf{z} = \mathbf{Bx}$ as follows:

$$\text{minimize} \quad \mathbf{c}^T\mathbf{B}^{-1}\mathbf{z} \quad \text{subject to} \quad \mathbf{AB}^{-1}\mathbf{z} = \mathbf{b}, \mathbf{z} \geq \mathbf{0}. \tag{13}$$

Similarly, we can define the LP problem Ψ_K from Φ_K in Eq. (5) via the transformation $\mathbf{w} = \mathbf{B}'\mathbf{y}'$ as:

$$\text{minimize} \quad \mathbf{c}'^T\mathbf{B}'^{-1}\mathbf{w} \quad \text{subject to} \quad \mathbf{A}'\mathbf{B}'^{-1}\mathbf{w} = \mathbf{b}', \mathbf{w} \geq \mathbf{0}. \tag{14}$$

Then, the underlying connections between Φ and Φ_K can be captured by the following theorem:

Theorem 4: *For any original LP problem Φ and its transformed problem Φ_K derived via our proposed transformation mechanisms in Section 3.4, the two related transformed problems Ψ and Ψ_K share the same feasible region.*

To prove Theorem 4, we first prove the following lemma. Note that hereafter we use $\mathbf{I}$ to denote the identity matrix for different dimensions, and assume the matrix dimensions always agree without further notice.

Lemma 1: For any $m \times n$ matrix $\mathbf{S}$ with full row rank m, $m \leq n$, and any $n \times m$ matrix $\boldsymbol{\lambda}$, if $|\mathbf{I} - \boldsymbol{\lambda}\mathbf{S}| \neq 0$, then $|\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}| \neq 0$.

Proof: Since $(\mathbf{I} - \boldsymbol{\lambda}\mathbf{S})$ is a $n \times n$ matrix, we have

$$\mathbf{S}(\mathbf{I} - \boldsymbol{\lambda}\mathbf{S}) = \mathbf{S} - \mathbf{S}\boldsymbol{\lambda}\mathbf{S} = (\mathbf{I} - \mathbf{S}\boldsymbol{\lambda})\mathbf{S}.$$

If we compute the rank of the matrices on both sides, denoted by $rank(\cdot)$, we have

$$rank(\mathbf{S}(\mathbf{I} - \boldsymbol{\lambda}\mathbf{S})) \quad = \quad rank((\mathbf{I} - \mathbf{S}\boldsymbol{\lambda})\mathbf{S})$$

Because $|\mathbf{I} - \boldsymbol{\lambda}\mathbf{S}| \neq 0$, from the left hand side we have

$$rank(\mathbf{S}(\mathbf{I} - \boldsymbol{\lambda}\mathbf{S})) = rank(\mathbf{S}) = m.$$

On the RHS, using basic linear algebra property we have

$$\begin{aligned} m = rank((\mathbf{I} - \mathbf{S}\boldsymbol{\lambda})\mathbf{S}) &\leq \min(rank(\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}), rank(\mathbf{S})) \\ &= \min(rank(\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}), m) \end{aligned}$$

Thus, $rank(\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}) = m$. Since $\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}$ is an $m \times m$ square matrix, it is indeed invertible, i.e., $|\mathbf{I} - \mathbf{S}\boldsymbol{\lambda}| \neq 0$. □

What Theorem 4 states is equivalent to prove $\mathbf{A}\mathbf{B}^{-1}\mathbf{x} = \mathbf{b}$ and $\mathbf{A}'\mathbf{B}'^{-1}\mathbf{w} = \mathbf{b}'$ have exactly the same solution set. Thus, all we need to do is to show the augmented matrix $[\mathbf{A}\mathbf{B}^{-1}\ \mathbf{b}]$ can be transformed to $[\mathbf{A}'\mathbf{B}'^{-1}\ \mathbf{b}']$ by left-multiplying an invertible matrix.

Proof: According to $\mathbf{A}', \mathbf{B}'$ in Eq. (3), we have

$$\begin{aligned} \mathbf{A}'\mathbf{B}'^{-1} &= \mathbf{Q}\mathbf{A}(\mathbf{B} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A})^{-1} \\ &= \mathbf{Q}\mathbf{A}[(\mathbf{I} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A}\mathbf{B}^{-1})\mathbf{B}]^{-1} \\ &= \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}(\mathbf{I} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A}\mathbf{B}^{-1})^{-1} \\ &= (\mathbf{I} - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda})^{-1}\mathbf{Q}\mathbf{A}\mathbf{B}^{-1}. \end{aligned}$$

The last equation is derived from Lemma 1 by viewing $\mathbf{Q}\mathbf{A}\mathbf{B}^{-1}$ as an $m \times n$ matrix with full row rank m. Thus,

$$\mathbf{Q}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda})\mathbf{A}'\mathbf{B}'^{-1} = \mathbf{A}\mathbf{B}^{-1}.$$

On the other hand, according to Eq. (3) and (4), we have $\mathbf{b}' = \mathbf{Q}(\mathbf{b} + \mathbf{A}\mathbf{r})$ and $\boldsymbol{\lambda}\mathbf{b}' = \mathbf{B}\mathbf{r}$. Therefore,

$$\begin{aligned} \mathbf{Q}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda})\mathbf{b}' &= \mathbf{Q}^{-1}(\mathbf{b}' - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda}\mathbf{b}') \\ &= \mathbf{Q}^{-1}(\mathbf{b}' - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\mathbf{B}\mathbf{r}) \\ &= \mathbf{Q}^{-1}\mathbf{b}' - \mathbf{A}\mathbf{r} = \mathbf{b}. \end{aligned}$$

Hence, the $m \times m$ random invertible matrix $\mathbf{Q}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda}) = (\mathbf{Q}^{-1} - \mathbf{A}\mathbf{B}^{-1}\boldsymbol{\lambda})$ is the row transformation matrix we are look for. Thus, it proves that $\mathbf{A}\mathbf{B}^{-1}\mathbf{z} = \mathbf{b}$ and $\mathbf{A}'\mathbf{B}'^{-1}\mathbf{w} = \mathbf{b}'$ have the same solution set. □

Remark. From Theorem 4, we know that an adversary can learn the feasible region on Ψ from Ψ_K. However, he cannot learn further information on Φ from Ψ due to the unknown value of $\mathbf{B}$, which serves for the transformation key between Φ and Ψ, and the random choice of $\boldsymbol{\lambda}, \mathbf{Q}, \mathbf{r}$. In particular, the objective function of Ψ is $\mathbf{c}^T\mathbf{B}^{-1}$, while the objective function of Ψ_K is $\mathbf{c}'^T\mathbf{B}'^{-1} = \gamma\mathbf{c}^T(\mathbf{B} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A})^{-1} = \gamma\mathbf{c}^T\mathbf{B}^{-1}(\mathbf{I} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A}\mathbf{B}^{-1})^{-1}$. Thus, as long as $\boldsymbol{\lambda}$ and $\mathbf{Q}$ can be set freely and kept secret together with $\gamma, \mathbf{B}, \mathbf{A}$, solving Ψ_K does not give adversary any advantage learning either the objective value or the objective function from Ψ, let alone the corresponding information on Φ. Moreover, since we have $\mathbf{x} = \mathbf{M}\mathbf{y} - \mathbf{r} = \mathbf{M}\mathbf{B}'^{-1}\mathbf{w} - \mathbf{r} = (\mathbf{B} - \boldsymbol{\lambda}\mathbf{Q}\mathbf{A})^{-1}\mathbf{w} - \mathbf{r}$, the secrecy of output is also protected well by the random choice of $\boldsymbol{\lambda}, \mathbf{Q}, \mathbf{r}$ and unknown $\mathbf{B}$. Thus, the aforementioned security analysis in Section 4.2 still holds.

5.2 Enhancements on Feasible Region Protection

While the above analysis is true for $\mathbf{B}$ as general invertible matrix as we assumed throughout this paper, there might be an issue for the special case of $\mathbf{B} = \mathbf{I}$. Because when $\mathbf{B} = \mathbf{I}$, Φ and Ψ becomes equivalent to each other. Thus, the adversary in possession of Ψ_K may know the feasible region of original problem Φ through Ψ directly.

To avoid this unwanted information leakage on feasible region, we propose to use an additional generalized permutation matrix (from products of non-singular diagonal and permutation matrices) $\mathbf{P}$ with positive[1] non-zero elements to multiply both sides of $\mathbf{B}'\mathbf{y} \geq \mathbf{0}$ in Eq. (3) and get $\mathbf{B}'' = \mathbf{P}\mathbf{B}'\mathbf{y} \geq \mathbf{0}$, which is now our final transformed inequality constraints. By doing so, the constraint of Ψ_K is now changed to $\mathbf{A}'\mathbf{B}''^{-1}\mathbf{w} = \mathbf{A}'\mathbf{B}'^{-1}\mathbf{P}^{-1}\mathbf{w} = \mathbf{b}', \mathbf{w} \geq \mathbf{0}$. Due to the random choice of $\mathbf{P}$, Ψ_K's feasible region is no longer the same as Ψ in Eq. (13) any more. That is, $\mathbf{A}\mathbf{B}^{-1}\mathbf{x} = \mathbf{b}$ and $\mathbf{A}'\mathbf{B}'^{-1}\mathbf{P}^{-1}\mathbf{w} = \mathbf{b}'$ no longer have the same solution set. Note that $\mathbf{P}$ does not have to be introduced for every transformation but only at this special case for enhanced protection of feasible region. Since $\mathbf{P}$ is a generalized permutation matrix, multiplying $\mathbf{P}$ to $\mathbf{B}'$ only incurs $O(n^2)$ cost, which can indeed be deemed lightweight, compared to other $O(n^3)$ matrix-matrix multiplications incurred in the problem transformation design.

6 Performance Analysis

6.1 Theoretic Analysis

6.1.1 Customer Side Overhead

For the three customer side algorithms KeyGen, ProbEnc, and ResultDec, it is straight-forward that

1. In fact, if $\mathbf{P}$ has negative values, then not all the rows in $\mathbf{P}\mathbf{B}'\mathbf{y}$ will remain larger than 0. Though it won't affect our result, for ease of presentation, we refine non-zero elements in $\mathbf{P}$ to be positive.

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 $O(n^{2.81})$ [20] or the Coppersmith-Winograd algorithm [21] in $O(n^{2.376})$. In either case, the customer side efficiency can be further improved.

6.1.2 *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 3.5). 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.

6.2 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 (http://www.mosek.com) through its Matlab interface to solve the original LP problem Φ and encrypted LP problem Φ_K. Our randomly generated test benchmark covers the small and medium sized problems, where m and n are increased from 50 up to 6,400 and 60 up to 7,680, respectively. The size of the benchmarks range from 51KB to 808.5MB. Both customer and cloud server computations in our experiment are conducted on the same workstation with an Intel Xeon X5650 CPU running at 2.66Ghz with 16 GB RAM. There are intentional design considerations that lead us to establish this experiment setup. But most importantly, it allows us to focus on the fair evaluation of computational efficiency of our proposed mechanism without worrying about communication latency, since the computational saving is one of our design goals and the computation dominates the running time as evidenced by our experiments and estimations in Section 6.2.1.

6.2.1 *Results on Normal LP problems*

Table 1 gives our experimental results, where each entry represents the mean of 20 trials. All these benchmarks are for the normal cases with feasible optimal solutions.

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 1 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 $50\times$ savings when the sizes of the original LP problems are not too small. As noted in Section 6.1.1, the customer's computational savings will increase when the problem size goes larger. This is indeed the case as shown in the last four rows in Table 1. 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. This is consistent with our analysis in Section 6.1.2 where the complexity to solve the original problem and the encrypted problem shall be the same. However, because we are performing random transformation of the original problem, it is possible that time of solving the transformed the problem might contain certain variations, and sometimes might even be less than solving the original problem. This explains why sometimes the cloud efficiency can be larger than 1 in Table 1. But overall, it confirms that secure outsourcing LP in cloud computing is economically viable.

Effect of Internet latency. Note that Internet latency is not a major issue in our computational savings evaluation. Indeed, many cloud service providers have provided means to establish dedicated high-speed connections to significantly accelerate the data communication between customer premises

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

Benchmark		Original Problem	Encrypted Problem		Asymmetric Speedup	Cloud Efficiency
#	small and medium 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	0.348	0.383	0.0008	434.3 $\times$	0.910
2	m = 100, n = 120	0.430	0.424	0.002	215.0 $\times$	1.016
3	m = 200, n = 240	0.772	0.981	0.012	62.53 $\times$	0.787
4	m = 400, n = 480	3.048	2.801	0.046	67.04 $\times$	1.088
5	m = 800, n = 960	14.220	15.097	0.276	51.52 $\times$	0.942
6	m = 1,600, n = 1,920	104.898	126.688	1.604	65.41 $\times$	0.830
7	m = 3,200, n = 3,840	873.610	1,117.148	9.418	92.76 $\times$	0.782
8	m = 6,400, n = 7,680	7,257.503	9,599.868	69.742	104.06 $\times$	0.756

and cloud, such as Amazon Direct Connect (http://aws.amazon.com/directconnect). The service is very economical, charing 0.3 USD/hour for 1Gbps dedicated connection. When applied to our benchmarks in Table 1, the transmission delay (excluding the data independent propagation delay) in sec is 0.00042, 0.0016, 0.0065, 0.0259, 0.1000, 0.4141, 1.6560, and 6.6237 for benchmark #1 to #8. Taking such latency as customer's local waiting time, the asymmetric speedup can still be 285.76×, 118.18×, 41.73×, 42.39×, 37.82×, 51.98×, 78.89×, 95.04×, respectively. We note that as the problem size m, n increase, the problem solving time at cloud increases much faster (at least $O(n^3)$) than the Internet latency (in order of $O(n^2)$). Thus, we can still expect to see practical computational savings. Finally, to further accelerate the data transmission, one alternative approach is to ship a batch of large transformed problems physically instead of electronically to the cloud, e.g. via Amazon's AWS Import/Export service (http://aws.amazon.com/importexport/), which has been considered faster and more cost effective.

We remark that for space interest, our experiments only focus on the proposed designs in Section 3. For the special case protection described in Section 5.2, the customer's extra cost on transformation, i.e., by randomly scaling the rows in $\mathbf{B}'$, is indeed lightweight and does not affect much on the overall computational savings. For example, when $m = 6400$, $n = 7680$, such extra transformation cost is no more than than 10 seconds in our experiments. As a result, the computational savings might be slightly lower due to the extra cost, but the overall practical efficiency still remains.

6.2.2 *Results on Infeasible LP problems*

In addition to normal cases, we also evaluate our results on the case of the infeasible LP problems. As discussed in Section 3.5.2, to verify the infeasible LP problem can eventually be reduced to the verification of its auxiliary LP problem with a non-zero optimal solution. One such example is shown in Eq. (8). Table 2 summarises the efficiency evaluations for this case. Similar to the normal case, these results are from random test benchmarks covering the small and medium sized problems. Here $t_{original}$ denotes the solving time for the original auxiliary LP problem, and t_{cloud} denotes the solving time for the encrypted auxiliary LP, which can easily be formulated by cloud after receiving the randomly transformed problem from customer. $t_{customer}$ represents the time for the problem transformation and result verification. From the table, we can clearly see the similar computational savings as derived in the normal case, this is primarily because these benchmarks are with the similar size of normal LP problems. In particular, we can almost achieve at least 40× savings, and the saving becomes larger when the problem size increases.

6.2.3 *Results on Unbounded LP problems*

Finally, we evaluate our results on the case of the unbounded LP problems. Similar to the normal case, we generate random test benchmarks that cover the small and medium sized problems, as shown in Table 3. Again, $t_{original}$, t_{cloud}, and $t_{customer}$ denotes the same physical meanings as in the infeasible case before. The timing results are obtained by operating over the auxiliary problems for the dual LP problems. Note that we intentionally use very small number of equality constraints m, as it allows us to more easily create our unbounded problem benchmarks. This also results in the faster solving process for both the original problem and the encrypted problem. But the computational savings, i.e., more than 60× savings, and cloud side efficiency are still in line with the normal and infeasible cases. This is because our transformation doesn't change the complexity of solving an LP, regardless of its being normal, infeasible, or unbounded. And the LP solver doesn't need to change its solving process regardless of its being an original or encrypted unbounded LP problem.

7 Related Work

7.1 Work on Secure Computation Outsourcing

General secure computation outsourcing that fulfills all aforementioned requirements, such as input/output

TABLE 2: Performance Results on Infeasible Case. Here $t_{original}$, t_{cloud}, and $t_{customer}$ denote the same physical meaning as shown in Table 1 and are measured by operating over the auxiliary LP problems.

Benchmark		Original Problem	Encrypted Problem		Asymmetric Speedup	Cloud Efficiency
#	small and medium 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	0.291	0.378	0.0008	364.6 ×	0.770
2	m = 100, n = 120	0.3025	0.3769	0.0025	121.8 ×	0.803
3	m = 200, n = 240	0.5913	0.600	0.0099	59.68 ×	0.986
4	m = 400, n = 480	2.207	2.538	0.056	39.24 ×	0.870
5	m = 800, n = 960	13.331	17.524	0.282	47.33 ×	0.760
6	m = 1,600, n = 1,920	184.898	232.576	1.416	130.57 ×	0.795
7	m = 3,200, n = 3,840	1,157.530	1,553.760	10.602	109.18 ×	0.745

TABLE 3: Performance Results on Unbounded Case. Here $t_{original}$, t_{cloud}, and $t_{customer}$ denote the same physical meaning as shown in Table 1 and are measured by operating over the auxiliary problem for the dual LP problems.

Benchmark		Original Problem	Encrypted Problem		Asymmetric Speedup	Cloud Efficiency
#	small and medium size	$t_{original}$ (sec)	t_{cloud} (sec)	$t_{customer}$ (sec)	$\frac{t_{original}}{t_{customer}}$	$\frac{t_{original}}{t_{cloud}}$
1	m = 10, n = 60	0.267	0.286	0.0005	497.88 ×	0.935
2	m = 20, n = 120	0.273	0.303	0.0014	189.62 ×	0.900
3	m = 40, n = 240	0.443	0.498	0.006	76.92 ×	0.889
4	m = 80, n = 480	1.446	1.991	0.018	81.13 ×	0.726
5	m = 160, n = 960	6.110	8.341	0.089	68.56 ×	0.732
6	m = 320, n = 1,920	45.322	46.575	0.479	94.56 ×	0.973
7	m = 640, n = 3,840	345.162	453.498	3.263	105.80 ×	0.761

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 [22] and/or oblivious transfer, 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.

7.2 Work on Secure Multiparty Computation

Another large existing list of work that relates to (but is also significantly different from) ours is Secure Multiparty 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, but the method only applies to the case where one party holds the cost function, and the other holds the constraints. 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, both Du [29] and Vaidya [30] have studied using disguising matrix based transformation approaches to tackle privacy-preserving linear programming problems. However, as later pointed out by Bednarz et al. [31], both Du's and Vaidya's approaches have correctness flaws, which may lead to returned solutions falling into infeasible region of original problems. To fix the problem, Bednarz et al. [31] propose to use generalized permutation matrices with only positive elements to disguise the linear constraints. However, such permutation matrices explicitly preserve the number of zero elements (aka. sparsity) of both the original constraint matrix and the original problem solution. Thus the input/ouput protection is not complete. Note that this is not the case in our generalized affine mapping based approach (see Section 4.2). Very recently, Mangasarian proposes two privacy-preserving formulations of linear programming over vertically [32] and horizontally partitioned [33] constraint matrix, respectively, among different involved entities. Both approaches are designed under SMC model and do not support a way to guarantee the quality of final solution in case of maliciously adversaries. Additionally, in his horizontally partitioned problem setting, the proposed approach is limited to hiding equality constraints only, and leaves secrecy of output unprotected.

7.3 Work on Cheating Detection

Detecting the unfaithful behaviors for computation outsourcing is not an easy task, even without consideration of input/output privacy. Verifiable computation delegation, where a computationally weak customer can verify the correctness of the delegated computation results from a powerful but untrusted server without investing too much resources, has found great interests in theoretical computer science community. Some recent general result can be found in Goldwasser et al. [34]. In distributed computing and targeting the specific computation delegation of one-way function inversion, Golle et al. [35] propose to insert some pre-computed results (images of "ringers") along with the computation workload to defeat untrusted (or lazy) workers. Szada et al. [36] extend the ringer scheme and propose methods to deal with cheating detection of other classes of computation outsourcing, including optimization tasks and Monte Carlo simulations. In [37], Du. et al. propose a method of cheating detection for general computation outsourcing in grid computing. The server is required to provide a Merkle-tree-based commitment for the results it computed. With the commitment, the customer can then use a sampling approach to carry out the result verification (without re-doing much of outsourced work.)

However, all above schemes allow server actually see the data and result it is computing with, which is strictly prohibited in the cloud computing model for data privacy. Thus, the problem of result verification essentially becomes more difficult, when both input/output privacy is demanded. Our work leverages the duality theory of LP problem and effectively bundles the result verification within the mechanism design, with little extra overhead on both customer and cloud server.

Difference from Conference Version [1] Firstly, we add Section 3.1 to thoroughly justifying our design intuition and problem motivation. Secondly, we add Section 5 to thoroughly investigate the underlying connections between our original and transformed problems via rigorous mathematical reasoning. Further, we completely redo and extend the experiments. Finally, we add Section 7.3 in related work on data computation delegation and cheating detection, and also compare recent advancements on privacy-preserving LP with ours.

8 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 random 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.

Acknowledgement

This work was supported by the US NSF under grant CNS-1262277 and CNS-1116939, RGC of Hong Kong (Project No. CityU 138513), and CityU of Hong Kong (Project No. 7200320).

References

[1] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. of IEEE INFOCOM*, 2011, pp. 820–828.
[2] P. Mell and T. Grance, "The nist definition of cloud computing," Referenced on Nov. 23rd, 2013 Online at http://csrc.nist.gov/publications/PubsSPs.html#800-145, 2011.
[3] Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing," 2009, online at http://www.cloudsecurityalliance.org.
[4] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.
[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, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE TPDS*, vol. 23, no. 8, pp. 1467–1479, 2012.
[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*, 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] C. Jansson, "An np-hardness result for nonlinear systems," *Reliable Computing*, vol. 4, no. 4, pp. 345–350, 1998.
[19] P. Van Hentenryck, D. McAllester, and D. Kapur, "Solving polynomial systems using a branch and prune approach," *SIAM Journal on Numerical Analysis*, vol. 34, no. 2, pp. 797–827, 1997.
[20] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, pp. 354–356, 1969.
[21] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. of STOC*, 1987, pp. 1–6.
[22] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. of EUROCRYPT*, 1999, pp. 223–238.
[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 Comp.*, 2006.
[26] T. Toft, "Solving linear programs using multiparty computation," in *Proc. of Financial Cryptography*, 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.
[29] W. Du, "A study of several specific secure two-party computation problems," Ph.D. dissertation, Purdue University, Indiana, 2001.
[30] J. Vaidya, "Privacy-preserving linear programming," in *Proc. of 24th ACM Symposium on Applied Computing*, 2009.
[31] 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.
[32] O. L. Mangasarian, "Privacy-preserving linear programming," *Optimization Letters*, vol. 5, pp. 165–172, 2011.
[33] ——, "Privacy-preserving horizontally partitioned linear programs," *Optimization Letters*, vol. 6, no. 3, pp. 431–436, 2012.
[34] S. Goldwasser, Y. Kalai, and G. Rothblum, "Delegating computation: interactive proofs for muggles," in *Proc. of STOC*, 2008.
[35] P. Golle and I. Mironov, "Uncheatable distributed computations," in *Proc. of CT-RSA*, 2001, pp. 425–440.
[36] D. Szajda, B. G. Lawson, and J. Owen, "Hardening functions for large scale distributed computations," in *Proc. of IEEE Symposium on Security and Privacy*, 2003, pp. 216–224.
[37] W. Du, J. Jia, M. Mangal, and M. Murugesan, "Uncheatable grid computing," in *Proc. of ICDCS*, 2004, pp. 4–11.

Cong Wang is an assistant professor in the Computer Science Department at City University of Hong Kong, Hong Kong. He received his B.E. and M.E. degrees from Wuhan University in 2004 and 2007, and Ph.D. degree from Illinois Institute of Technology in 2012, all in electrical and computer engineering. He interned at the Palo Alto Research Center in the summer of 2011. His research interests are in the areas of cloud computing security, with current focus on secure data outsourcing and secure computation outsourcing in public cloud.

Kui Ren is an associate professor of Computer Science and Engineering and the director of UbiSeC Lab atState University of New York at Buffalo. He received his PhD degree from Worcester Polytechnic Institute. Kui's current research interest spans Cloud & Outsourcing Security, Wireless & Wearable System Security, and Human-centered Computing.His research has been supported by NSF, DoE, AFRL, MSR, and Amazon. He is a recipient of NSF CAREER Award in 2011 and Sigma XiIIT ResearchExcellence Award in 2012. Kui has published 135 peer-review journal and conference papers and received several Best Paper Awards including IEEEICNP 2011. He currently serves as an associate editor for IEEE TMC, TIFS, IEEE Wireless Communications,IEEE IoT-J, IEEE TSG, Elsevier Pervasive and MobileComputing, and Oxford The Computer Journal. Kui is a senior member of IEEE, a member of ACM, and a Distinguished Lecturer of IEEE VehicularTechnology Society.

Jia Wang received the B.S. degree in Electronic Engineering from Tsinghua University, Beijing, China, in 2002, and the Ph.D. degree in Computer Engineering from Northwestern University, Evanston, IL, in 2008. He is an Associate Professor of Electrical and Computer Engineering at Illinois Institute of Technology, Chicago, IL. His research interests are on algorithm design for emerging computing platforms, with a focus for computer-aided design of very large scale integrated circuits.