Practical and Secure Outsourcing of Linear Programming in Cloud Computing

K. Jaya Santhosh, S. Reshma
Department of C.S.E., CR Engineering College/JNTU Anantapur

Abstract— Despite the tremendous benefits in Cloud Computing, security is the primary obstacle, especially for customers when their confidential data are consumed and produced during the computation. Treating the cloud as an intrinsically insecure computing platform from the viewpoint of the cloud customers, we must design mechanisms that not only protect sensitive information by enabling computations with encrypted data, but also protect customers from malicious behaviors by enabling the validation of computation result.

This paper investigates secure outsourcing of widely applicable linear programming (LP) computations. To achieve practical efficiency, our mechanism design decomposes the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. In particular, original LP problem is transformed into some arbitrary one while protecting sensitive input/output information. To validate the computed 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 design.

Index Terms— *Cloud Computing, Multiparty Computation, Problem Transformation, Secure Outsourcing.*

I. INTRODUCTION

Cloud Computing has great potential of providing robust computational power to the society at reduced cost. It 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. 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 purchasing hardware and software and/or the operational overhead therein.

Outsourcing computation to the commercial public cloud deprives 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 proprietary research data, business financial records, 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 exist various motivations for cloud server to behave unfaithfully and at worse 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.

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 circuit that allows to be evaluated with encrypted private inputs. 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. 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

Manuscript received Dec 20, 2012.

Mr. K. Jaya Santhosh, M.Tech Student, Department of CSE, Chadalawada Ramanamma Engg. College/JNTU Anantapur, Andhra Pradesh, India.
Mobile No.: +91 9959326228

Mrs. S. Reshma, Assistant Professor, Department of CSE, Chadalawada Ramanamma Engg. College/JNTU Anantapur, Andhra Pradesh, India. Mobile No.: +91 9703398494

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 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.

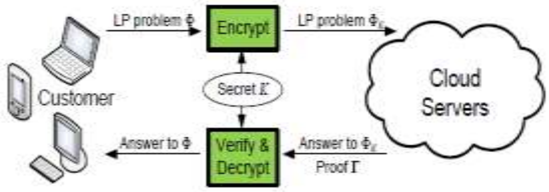

Fig.1: Architecture of secure outsourcing linear programming problems in Cloud Computing.

The rest of the paper is organized as follows. The Problem Statement which depicts the System and Threat model, Design goals and Background on LP is shown in Section II. Section III deals the Proposed Schemes. Section IV furnishes the Security Analysis. Section V describes the Performance Analysis. Section VI overviews the related work. Finally, Section VII gives the Concluding remarks.

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 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 semihonest 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. Finally note that we assume the communication channels between each cloud server and the customer is authenticated and reliable, which can be achieved in practice with little overhead.

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

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. 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,

$$\text{Minimize } p^T x \quad \text{subject to} \quad Qx = r,\ x \geq 0. \tag{1}$$

Here x is an nx1 vector of decision variables. Q is an mxn matrix, and both p and r are nx1 vectors. It can be assumed further that $m \leq n$ and that Q has full row rank; otherwise, extras rows can always be eliminated from Q.

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

$$\text{Minimize } \quad p^T x \quad \text{subject to} \quad Qx = r,\ Rx \geq 0. \tag{2}$$

In Eq. (2), we replace the non-negative requirements in Eq. (1) by requiring that each component of Rx to be non-negative, where R is an nxn non-singular matrix, i.e. Eq. (2) degenerates to Eq. (1) when R is the identity matrix. Thus, the LP problem can be defined via the tuple $\Phi = (Q, R, r, p)$ as input, and the solution x as output.

III. PROPOSED SCHEME

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. Our LP outsourcing scheme provide a complete outsourcing solution for not only the privacy protection of problem input/output, but also its efficient result checking.

A. Mechanism Design Framework

We propose to apply problem transformation for mechanism design. The general framework is adopted from a generic approach, 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.

1. **KeyGen**(1^k) → {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.
2. **ProbEnc**(K, Φ) → { Φ_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.
3. **ProofGen**(Φ_K) → {(y, Γ)}. This algorithm augments a generic solver that solves the problem Φ_K to produce both the output y and a proof Γ. The output y later decrypts to x, and Γ is used later by the customer to verify the correctness of y or x.
4. **ResultDec** (K, Φ, y, Γ) → {x, Ψ}. This algorithm may choose to verify either y or x via the proof Γ. In any case, a correct output x is produced by decrypting y using the secret K. The algorithm outputs Ψ when the validation fails, indicating the cloud server was not performing the computation faithfully.

Note that our proposed mechanism provides us one-time pad types of flexibility. Namely, we shall never use the same secret key K to two different problems. Thus, when analyzing the security strength of the mechanism, we focus on the ciphertext only attack. We do not consider known plain text attack in this paper but do allow adversaries to do offline guessing or inferring via various problem-dependent information including sizes and signs of the solution, which are not necessary to be 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 (Q, r)*:* First of all, a randomly generated m × m non-singular matrix Q can be part of the secret key K. The customer can apply the matrix to Eq. (2) for the following constraints transformation,

$$Qx = r \qquad \Rightarrow \qquad Q'x = r'$$

Where $Q' = AQ$ and $r' = Ar$. Since we have assumed that Q has full row rank, Q′ must have full row rank. Without knowing A, it is not possible for one to determine the exact elements of Q. However, the null spaces of Q and Q′ remain the same, which may violate the security requirement of some applications. The vector r is encrypted in a perfect way since it can be mapped to an arbitrary r′ with a proper choice of A.

2) Hiding inequality constraints (R)*:* The customer cannot transform the inequality constraints in the similar way as used for the equality constraints. This is because for an arbitrary invertible matrix A, $Rx \geq 0$ is not equivalent to $ARx \geq 0$ in general. To hide R, 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.

$$\begin{array}{lcl} Qx = r & \Rightarrow & Qx = r \\ Rx \geq 0 & & (R - \lambda Q)x = R'x \geq 0 \end{array}$$

where λ is a randomly generated n x m matrix in K satisfying that $|R'| = |R - \lambda Q| \neq 0$ and $\lambda r = 0$. Since the condition $\lambda r = 0$ is largely underdetermined, it leaves great flexibility to choose λ in order to satisfy the above conditions.

3) Hiding objective functions p *and value* p^Tx *:* Given the widely application of LP, such as the estimation of business annual revenues or personal portfolio holdings etc., the information contained in objective function p and optimal objective value p^Tx might be as sensitive as the constraints of Q, R, r. 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 p is replaced by γp. It is not possible to derive the original optimal objective value p^Tx 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 p. The number and position of zero-elements in p are not protected. Besides, the ratio between elements in p is also preserved after constant scaling.

Summarization of basic techniques: Overall, the basic techniques would choose a secret key K = (A, λ, γ) and encrypt the input tuple Φ into $\Phi_K = (Q', R', r', p)$, 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 x during the problem input encryption. We propose to encrypt the feasible region of Φ by applying an affine mapping on the decision variables 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 M be an nxn non-singular matrix and r be an nx1 vector. The affine mapping defined by M and r transforms x into $y = M^{-1}(x + f)$. 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 y.

$$\begin{array}{ll} \text{Minimize} & p^TMy - p^T f \\ \text{Subject to} & QMy = r + Qf \\ & RMy \geq Rf. \end{array}$$

Using the basic techniques, this LP problem can be further transformed to

$$\begin{array}{ll} \text{Minimize} & \gamma p^TMy \\ \text{Subject to} & QAMy = A(r + Qf), \\ & RMy - \lambda AQMy \geq Rf - \lambda A(r + Qf). \end{array}$$

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

$$\begin{aligned} Q' &= AQM \\ R' &= (R - \lambda AQ)M \\ r' &= A(r + Qf) \\ p' &= \gamma M^T p \end{aligned} \tag{3}$$

If the following conditions hold,

$$|R'| \neq 0,\ \lambda r' = Rf, \text{ and } r + Qf \neq 0 \tag{4}$$

then the LP problem $\Phi_K = (Q', R', r', p')$ can be formulated via Eq. (5),

$$\text{Minimize} \quad p'^Ty \quad \text{subject to} \quad Q'y = r',\ R'y \geq 0. \tag{5}$$

Discussion: By keeping the randomly selected M and 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. In many cases, especially when the computation on the cloud requires a huge amount of computing resources, there exist 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 $\Phi_K = (Q', R', r', p')$, we propose to solve the result verification problem by designing a method to verify the correctness of the solution 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. 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. We first assume that the cloud server returns an optimal solution y. In order to verify 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,

$$\text{Maximize} \quad r'^T s \quad \text{subject to} \quad Q'^T s + R'^T t = p', \ t \geq 0, \qquad (6)$$

Where *s* and *t* are the mx1 and nx1 vectors of dual decision variables respectively. The strong duality of the LP problems states that if a primal feasible solution y and a dual feasible solution (s, t) lead to the same primal and dual objective value, then both y and (s, 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 y can be verified based on the following conditions,

$$p'^T y = r'^T s, \ \ Q'y = r', \ \ R'y \geq 0, \ \ Q'^T s + R'^T t = p', \ \ t \geq 0. \quad (7)$$

Here, $p'^T y = r'^T s$ tests the equivalence of primal and dual objective value for strong duality. All the remaining conditions ensure that y and (s, 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 $Q'y = r'$ can be achieved in practice by checking whether $\|Q'y - r'\|$ is small enough.

IV. SECURITY ANALYSIS

A. *Analysis on Correctness and Soundness Guarantee*

We give analysis on correctness and soundness guarantee via 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 y computed by honest cloud server will always be verified successfully. This follows directly from the duality theorem of linear programming. Namely, all conditions derived from duality theorem and auxiliary LP problem constructions for result verification are necessary and sufficient. Next, we show that correctly verified solution y always corresponds to the optimal solution 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 contraction, suppose $x = My - f$ is not the optimized solution for Φ. Then, there exists x* such that $p^T x^* < p^T x$, where $Qx^* = r$ and $Rx^* \geq 0$. Since $x^* = My^* - f$, it is straightforward that

$$p^T My^* - p^T f = p^T x^* < p^T x = p^T My - p^T f$$

Where $Q'y^* = r'$ and $R'y^* \geq 0$. Thus, y* is a better solution than y for problem Φ_K, which contradicts the fact that the optimality of y has been correctly verified. This completes the proof of **theorem** 1.

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 and output privacy guarantee. Note that the only information that the cloud server obtains is $\Phi_K = (Q', R', r', p')$. We start from the relationship between the primal problem Φ and its encrypted one Φ_K. First of all, the matrix Q and the vector r are protected perfectly. Secondly, the information of matrix R is protected. Recall that the nxm matrix λ in the condition $\lambda r' = Rf$ is largely underdetermined. Thirdly, the vector p is protected well by scaling factor γ and M.

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,

$$\text{Maximize} \quad r^T\alpha \quad \text{subject to} \quad Q^T\alpha + R^T\beta = p, \ \beta \geq 0 \qquad (8)$$

Where α and β are the mx1 and nx1 vectors of dual decision variables respectively.

Clearly, the analysis for primal problem Φ's input privacy guarantee still holds for its dual problem input (Q, R, r, p). As for the output privacy, we plug eq. (3) into Φ_K's dual problem defined in eq. (6) and rearrange it as,

$$\begin{aligned} &\text{Maximize} \quad [A(r + Qf)]^T s \\ &\text{Subject to} \quad Q^T A^T (s - \lambda^T t) + R^T t = \gamma p, \quad t \geq 0, \qquad (9) \end{aligned}$$

Note that M^T in the equality constraint is canceled out during the rearrangement. Comparing eq. (8) and eq. (9), we derive the linear mapping between (α, β) and (s, t) as,

$$\alpha = [A^T (s - \lambda^T t)]/\gamma, \quad \beta = t/\gamma \qquad (10)$$

V. PERFORMANCE ANALYSIS

A. Theoretic Analysis

1) Customer Side Overhead: According to our mechanism, customer side computation overhead consists of key generation, problem encryption operation, and result verification, which corresponds to the three algorithms KeyGen, ProbEnc, and ResultDec, respectively. Because KeyGen and Result-Dec only require a set of random matrix generation as well as vector-vector and matrix-vector multiplication, the computation complexity of these two algorithms are upper bounded via $O(n^2)$. Thus, 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 $O(n^\rho)$ for some $2 < \rho \leq 3$. In our paper, 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 [18] with time complexity $O(n^{2.81})$ or the Coppersmith-Wino grad algorithm [19] in $O(n^{2.376})$. In either case, the overall 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 ProofGen. The cloud server just solves the encrypted LP problem Φ_K with the dual optimal solution as the result proof Γ, which is usually readily available and incurs no additional cost for cloud. 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 process 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.

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 experiment 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 $t_{original}/t_{customer}$, 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 $t_{original}/t_{cloud}$, 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 30x 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.

TABLE 1: 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 closer to 1 as possible.

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

Table 1: Experiment results.

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. In [5] they gave 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. Later on in [7] and [8] Atallah et al. gave 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 et al. [10] gave a provably secure protocol for secure outsourcing matrix multiplications based on secret sharing [23]. 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.

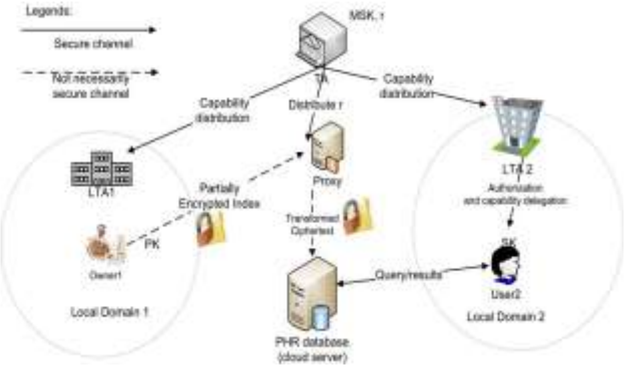

Fig.2: Secure outsourcing in Cloud Computing.

B. 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] and later extended by Goldreich et al. [24] and many others. 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. [25]. 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. Besides, in SMC no single involved party knows the entire problem input information, making result verification usually a difficult task. Recently, Li and Atallah [26] give a study for secure and collaborative computation of linear programming under the SMC framework. 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. This has the asymmetry issue mentioned previously. Besides, they only consider honest-but-curious model and thus do not guarantee that the final solution is optimal.

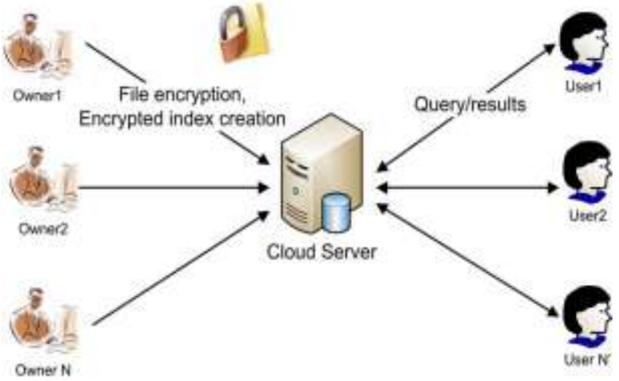

Fig.3: Multiparty computation in Cloud Computing.

C. Delegating Computation and 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 many resources, has found great interests in theoretical computer science community. Some recent general result can be found in Goldwasser et al. [27] In distributed computing and targeting the specific computation delegation of one-way function inversion, Golle et al. [28] propose to insert some pre-computed results (images of "ringers") along with the computation workload to defeat untrusted (or lazy) workers. In [29], Du. et al. propose a method of cheating detection for general computation outsourcing in grid computing. The server is required to provide a commitment via a Merkle tree based on the results it computed. The customer can then use the commitment combined with a sampling approach to carry out the result verification (without re-doing much of the 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 input/output privacy is demanded.

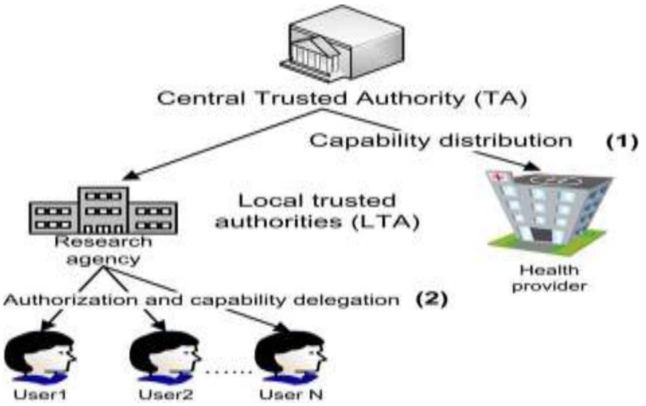
Central Trusted Authority (TA)
Capability distribution (1)
Local trusted authorities (LTA)
Research agency
Health provider
Authorization and capability delegation (2)
User1
User2
User N

Fig. 4: Delegation & cheating detection in Cloud Computing.

VII. CONCLUSION

In this paper, for the first time, we formalize the problem of securely outsourcing LP computations in cloud computing, and provide 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 develop problem transformation techniques that enable customers to secretly transform the original LP into some arbitrary one while protecting sensitive input/output information. We also investigate duality theorem and derive 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 demonstrate the immediate practicality of the proposed mechanism.

We plan to investigate some interesting future work as to devise robust algorithms to achieve numerical stability; to explore the sparsity structure of problem for further efficiency improvement; to establish formal security framework and to extend our result to non-linear programming computation outsourcing in cloud.

ACKNOWLEDGMENT

We would like to thank our Head of the Department, Department of CSE; Principal and Chairman of CREC, Tirupati, for their strenuous efforts in bringing this paper into reality. We would be overwhelmed for the review and comments of the Editor-in-Chief, the Associate Editor and the anonymous Referees of IJARCET.

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. 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 6th 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 CRYPT'10*, Aug. 10.

[10] M. Atallah and K. Frikken, "*Securely Outsourcing Linear Algebra Computations*," in *Proc. of ASIACCS*, 2010, pp. 48–59.

[11] C.C.Yao, "*Protocols for Secure Computations (extended abstract)*," in *Proc. of FOCS'82*, 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'10*, 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*," *Numerical Mathematics.*, vol. 13, pp. 354–356, 1969.

[19] D. Coppersmith and S. Winograd, "*Matrix Multiplication via Arithmetic Progressions*," in *Proc. of STOC'87*, 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'99*, 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,'79.

[24] O. Goldreich, S. Micali, and A. Wigderson, "*How to play any mental game or a completeness theorem for protocols with honest majority*,", in *Proc. of STOC'87*, 1987, pp. 218–229.

[25] 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.

[26] J. Li and M. J. Atallah, "*Secure and Private Collaborative Linear Programming*" in *Proc. of CollaborateCom*, Nov. 2006.

[27] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "*Delegating Computation: Interactive Proofs for Muggles*" in *Proc. of STOC*, 2008, pp. 113–122.

[28] P. Golle and I. Mironov, "*Uncheatable Distributed Computations*," in *Proc. of CT-RSA*, 2001, pp. 425–440.

[29] W. Du, J. Jia, M. Mangal, and M. Murugesan, "*Uncheatable Grid Computing*," in *Proc. of ICDCS*, 2004, pp. 4–11.

AUTHORS PROFILE

Mr. K. JAYA SANTHOSH completed his B.Tech in CSE from Sri Venkateswara University, Tirupati, in 2003. He has 7 years of teaching experience. At present he is pursuing Masters Degree in Technology in CSE from Jawaharlal Nehru Technological University, Anantapur.
Mobile No.: +91 9959326228

Mrs. S. Reshma received her B.Tech & M.Tech degrees in CSE from Jawaharlal Nehru Technological University, Anantapur. Currently she is working as Assistant Professor in the Department of CSE at Chadalawada Ramanamma Engineering College, Tirupati. She has 4 years of teaching experience.
Mobile No.: +91 9703398494