International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

Outsourcing of Computations in Cloud Computing

B V Janaki Savitri[1], V Madhavi[2],
PG Scholar (M.Tech.) Assistant Professor
Department of Computer science and Engineering
Aurora's Technological and Research Institute,
Parvathapur, Uppal, Hyderabad, India

***Abstract*—Due to the availability of massive and scalable computational power economically, the emerging cloud computing paradigm has been attractive to the customers with limited computational resources to outsource their large computation workloads. However, security and privacy concerns are majorly obstructing the widespread adoption of this promising computing model especially when the confidential data of the customers is consumed and produced during the computations in the cloud. Devising a mechanism for general secure computation outsourcing was so far theoretically feasible and designing mechanisms that are practically efficient remains a very challenging problem. Focusing on engineering computing and optimization tasks, Cong Wang et al. developed a scheme for secure outsourcing of widely applicable linear programming (LP) computations in the cloud. Also, several works have discussed the outsourcing of nonlinear programming (NLP) computations. In this work we are intended to study and thoroughly analyse both LP and NLP computation outsourcing. Our experimental results show that, due to the complex computations involved, NLP computations consume more time, but, secure than the LP computations outsourcing comparatively.**

***Keywords*— Linear Programming, Non-linear Programming, Cloud computing, Security and privacy, Outsourcing**

I. INTRODUCTION

Cloud Computing involves on-demand access to a shared pool of configurable computing resources [1]. One of the key benefits of the cloud paradigm is computation outsourcing, in which the computational power of cloud customers is no longer limited by their resource-constraint devices. Regardless of the tremendous benefits, outsourcing computation to the commercial public cloud is also limiting the customer's direct control over the systems that consume and produce their data during the computation, which brings in new security concerns and challenges [2]. The outsourced computation workloads often contain sensitive information, such as the business financial records, proprietary research data, or personally identifiable health information etc. To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing [2] so as to provide end-to-end data confidentiality assurance in the cloud and beyond.

However, ordinary data encryption techniques prevent cloud from performing any meaningful operation of the underlying plaintext data [3], making the computation over encrypted data a very hard problem. Moreover, 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 return incorrect results. Besides, possible software bugs, hardware failures, or even outsider attacks might also affect the quality of the computed results. Thus, 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 fewer amounts of operations following the mechanism than completing the computations by themselves directly.

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 [6], [7], or multi-round interactive protocol executions [5], or huge communication complexities [8], 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, Cong Wang et al.[10] proposed an efficient mechanisms for secure outsourcing of linear programming (LP). Also the work by [11] reported the outsourcing techniques for nonlinear programming (NLP) computations. Linear and nonlinear programming methods are algorithmic and computational tools which captures the first order effects of various system parameters that should be optimized, and is essential to engineering optimization. They are widely used in various engineering disciplines that analyse and optimize real-world systems, such as packet routing, flow control, power management of data centres, etc. [9]. Because LP and NLP computations require a substantial amount of computational power and usually involve confidential data, it is proposed to explicitly decompose the LP and NLP

computation outsourcing into public LP and NLP solvers running on the cloud and private LP and NLP parameters owned by the customer. The flexibility of such decomposition allows exploring higher-level abstraction of LP and NLP computations than the general circuit representation for the practical efficiency. Fig1. shows the architecture for outsourcing of LP and NLP computations.

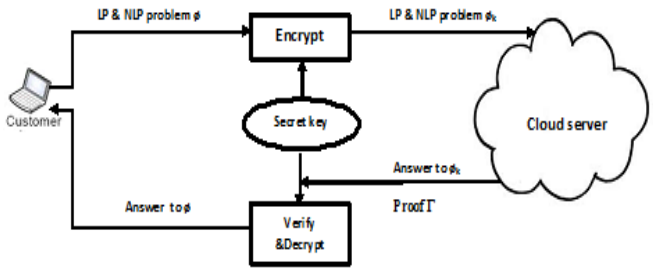

Fig. 1 The architecture for outsourcing of LP and NLP computations

Such methods of outsourcing should perform result validation, which must 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 and NLP problem.

A. Design Goals

To enable secure outsourcing of LP and NLP under the aforementioned model, the mechanism must attain the following design goals.

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 while performing the LP and NLP computations.
4) **Efficiency**: The local computations done by customer should be substantially less than solving the original LP and NLP on his own. The computation burden on the cloud server should be within the comparable time complexity of existing practical algorithms solving LP and NLP problems.

II. LINEAR PROGRAMMING

Usually, an optimization problem is formulated as a mathematical programming problem that needs the values for a set of decision variables to minimize or maximize an objective function to represent the cost subjected to a set of constraints. The objective function is an affine function of the decision variables and the constraints are a system of linear equations and inequalities for linear programming. A non-negative slack variable can be introduced to express a constraint in the form of linear equation, which is a linear inequality and the difference of two non-negative auxiliary variables can be expressed as a free decision variable.

Any linear programming problem can be expressed in the following standard form,

$$\text{Minimize } c^T x \text{ subject to } Ax = b,\ x \geq 0. \qquad (1)$$

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

A more general form of LP problem is as follows,

$$\text{Minimize } c^T x \text{ subject to } Ax = b,\ Bx \geq 0. \qquad (2)$$

Here, B is an n × n non-singular matrix, i.e. Eq. (2) degenerates to Eq. (1) when B is the identity matrix. Thus, the LP problem can be defined via the tuple $\Phi = (A,B, b, c)$ as input, and the solution x as output.

Several basic and enhanced techniques have been proposed by Wang et al.[10] to encrypt the problem parameters. The enhanced techniques employ affine mapping on the original problem in order to maintain privacy of the feasible region. In order to verify the result they have used the conditions derived from the duality property of linear programming. The result verification is done for all the three possible cases of the LP problem viz., Feasible, Infeasible and Unbounded. For general understanding purpose we give the basic techniques below. It should be noted that, the input encryption based on these techniques results in an unsatisfactory mechanism. However, the analysis will give insights on how a stronger mechanism should be designed. It is assumed that the cloud server honestly performs the computation, in order to simplify the presentation.

1) Hiding equality constraints (A, b): 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,

$$Ax = b \Rightarrow A'x = b'$$
$$\text{where } A' = QA \text{ and } b' = Qb$$

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

2) Hiding inequality constraints (B): 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 Q, $Bx \geq 0$ is not equivalent to $QBx \geq 0$ in general. To hide B, the fact that a feasible solution to Eq. (2) must satisfy the equality constraints, is applied. To be more specific, the feasible regions defined by the following two groups of constraints are the same.

$$\begin{cases} Ax = b \\ Bx \geq 0 \end{cases} \Longrightarrow \begin{cases} Ax = b \\ (B - \lambda A)x = B'x \geq 0 \end{cases}$$

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

3) Hiding objective functions c and value $c^T x$: 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 c and optimal objective value cT x might be as sensitive as the constraints of A,B, b. Thus, they should be protected, too.

To achieve this, constant scaling is applied to the objective function, i.e. a real positive scalar γ is generated randomly as part of encryption key K and c is replaced by γc. It is not possible to derive the original optimal objective value $c^T x$ 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 c. Namely, the number and position of zero-elements in c are not protected. Besides, the ratio between the elements in c is also preserved after constant scaling. Overall, the basic techniques would choose a secret key K = (Q, λ, γ) and encrypt the input tuple Φ into $\Phi_K = (A', B', b', \gamma c)$, which gives reasonable strength of problem input hiding. However, 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.

III. NONLINEAR PROGRAMMING

Nonlinear programming (NLP) involves minimizing or maximising a nonlinear objective function subject to bound constraints, linear constraints, or nonlinear constraints, where the constraints can be inequalities or equalities. Example problems in engineering include analysing design tradeoffs, selecting optimal designs, and incorporating optimization methods in algorithms and models. In NLP outsourcing scenario, [11] proposed a method to deal with the Sequential Quadratic programming method of NLP.

Usually, the Non-linear function f(x) is under the non-linear inequality constraints

$$x_k \in R^{n:} \min f(x),\ g(x) \leq 0 \qquad (3)$$

where x is an n-dimensional parameter vector. The vector-valued function g(x) defines m inequality constraints, $g(x)=(g_1(x),.....g_m(x))^T$. To simplify (3), the equality constraints and upper or lower bounds of the variables are omitted. The problem is considered to be non-convex and non-linear in general. Sequential Quadratic Programming is considered a best method to solve smooth Non-Linear Optimization problems by using standard general purpose algorithms. The following assumptions are made.

1. The problem is not too big.
2. Function values can be calculated within sufficient precision.
3. The problem is smooth and well scaled.

The sub-problems consist of strictly convex quadratic programming problems with inequality constraints obtained by linearizing the constraints and by approximating the Lagrangian function of (3) quadratically. The Sequential Quadratic Programming has the roots in unconstrained optimization. The main objective behind is to establish a quadratic approximation based on second order information with the goal to achieve a fast local convergence speed. The linearly constrained, strictly convex quadratic program must be solved in each iteration step by an available black box solver. This is mostly used in structural optimization. The method is based on the observation that in some special cases, typical structural constraints become linear in the inverse variables. Although this special situation is rarely observed in practice, a suitable substitution of structural variables by inverse ones depending on the sign of the corresponding partial derivatives and subsequent linearization is expected to linearize constraints somehow. To formulate the sub-problem, as said above, the process starts from the given iterates $x_k \in R^n$ an approximation of the solution, $u_k \in R^m$, an approximation of the vector of multipliers and $B_k \in R^{n\times n}$, an approximation of the Hessian of the Lagrange function. Then we obtain sub problem,

$$y \in R^n : \min f^k(y),\ g^k(y) \leq 0 \qquad (4)$$

by defining

$$f_k(y) = 1/2(y-x_k)^T B_k(y-x_k) + \nabla f(x_k)^T (y-x_k) + f(x_k)$$

$$g^k_j = \nabla g_j (x_k)^T (y-x_k) + g_j(x_k),\ j = 1,.....,m$$

It is clearly visible that the requirements of (4) are satisfied. The core idea is to approximate the second order information to get a fast final convergence speed. The most interesting feature of the Sequential Quadratic Programming method is the super linear convergence speed in the neighbourhood of the solution. That is,

$$\| x_{k+1} - x^* \| < \gamma_k \| x_k - x^* \| \quad \text{with } \gamma_k \rightarrow 0$$

In order to achieve practical efficiency, the mechanism design explicitly decomposes the NLP computation outsourcing into public NLP solvers running on the cloud and private NLP parameters owned by the customer. The resulting flexibility allows to explore appropriate security efficiency trade off via higher-level abstraction of NLP computations than the general circuit representation. Initially a framework is designed for the approach. After presenting the basic techniques the existing approach of the cloud computer is

being extended to the security strength of NLP outsourcing, it must be able to change the feasible region of original NLP and at the same time hide output vector x during the problem input encryption. Finally the proposed approach is evaluated for the security with the existing approaches.

IV. THE GENERAL DESIGN FRAMEWORK

The general framework is adopted from a generic approach [9], while the instantiation as per [10] 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.

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 or NLP problem.

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.

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.

ResultDec(K, Φ , y, Γ) → {x, ⊥}. 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 ⊥ when the validation fails, indicating the cloud server was not performing the computation faithfully.

This mechanism provides one-time-pad types of flexibility. The same secret key K is never used for two different problems.

V. EXPERIMENTAL RESULTS

An experimentation to compare the performance of Linear Programming over Non-Linear Programming in a Cloud Computing environment is performed. From the outcome it is apparent that the NLP outsourcing consumes more computational time compared to that of the LP outsourcing due to the internal security computations. Though, the computations may be complex, it is evident from the results that the NLP computation outsourcing is more secure compared to that of the LP outsourcing.

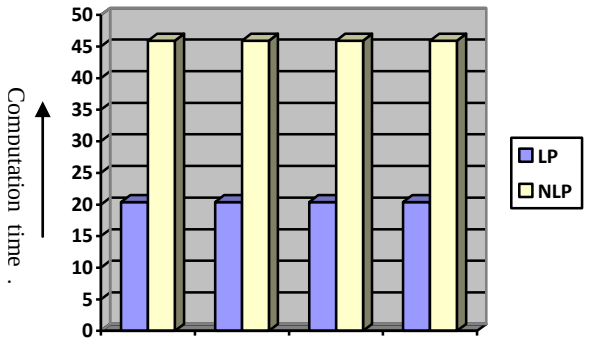

Fig. 2 Computation Time comparison LP Vs NLP

VI. CONCLUSIONS

In this paper a study of Linear and Nonlinear programming computations outsourcing in the cloud computing environment is carried out. Experimentation is carried out by implementing the basic schemes proposed in the literature. Sequential Quadratic Programming method is used to implement NLP scheme. From the results it is observed that LP programming computations involve less time than the NLP computations. Both LP and NLP mechanisms involve almost same communication complexity. Due to the complexity in operations and computations it is found that NLP computation outsourcing is more secure than LP outsourcing.

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. cloud securityalliance.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] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," Int. J. Inf. Sec., vol. 4, no. 4, pp. 277–287, 2005.
[7] 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.
[8] M. Atallah and K. Frikken, "Securely outsourcing linear algebra computations," in Proc. of ASIACCS, 2010, pp. 48–59.
[9] D. Luenberger and Y. Ye, Linear and Nonlinear Programming, 3rd ed. Springer, 2008.
[10] Cong Wang, Kui Ren, and Jia Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing", IEEE Transactions On Cloud Computing, April 10-15, 2011. pp. 1-9
[11] M Madhura, R Santosh, "An Efficient and Secure Nonlinear Programming Outsourcing in Cloud Computing", International Journal of Computer Applications (0975 – 8887) Volume 43– No.7, April 2012,pp.36-39