Linear Programming Concepts for Secure and Practical Outsourcing

Madesh Kumar S

***Abstract*— Internet computing technologies, like grid computing, enable a weak computational device connected to such a grid to be less limited by its inadequate local computational, storage, and bandwidth resources. However, such a weak computational device (PDA, smartcard, sensor, etc.) often cannot avail itself of the abundant resources available on the network because its data are sensitive [1]. Cloud computing promises greater flexibility in business planning along with significant cost savings by leveraging economies of scale in the IT infrastructure. It also offers a simplified capital and expenditure model for compute services as well as increased agility for cloud customers who can easily expand and contract their IT services as business needs change [2]. The main objective of cloud computing enables customers with limited computational resources to outsource their large computation workloads to the cloud, and economically enjoy the massive computational power, bandwidth, storage, and even appropriate software that can be shared in a pay-per-use manner. One fundamental advantage of the cloud paradigm is computation outsourcing, where the computational power of cloud customers is no longer limited by their resource-constraint devices [4].**

Outsourcing is to store the task or data which user wants it to do from outside their system such as cloud. By outsourcing the workloads into the cloud, customers could enjoy the literally unlimited computing resources in a pay-per-use manner without committing any large capital outlays in the purchase of hardware and software and/or the operational overhead therein[3].

From outsourced computation workloads often contain sensitive information. On the other hand, the operational details inside the cloud are not transparent enough to 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 results.

***Index Terms*—Internet computing technology, PDA, smartcard, sensor, secure computation outsourcing.**

I. INTRODUCTION

Large-scale problems in the physical and life sciences are being revolutionized by Internet computing technologies, like grid computing [10], that make possible the massive cooperative sharing of computational power, bandwidth, storage, and data. Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential **characteristics,** three **service models**, and four **deployment models**. Cloud computing promises greater flexibility in business planning along with significant cost savings by leveraging economies of scale in the IT infrastructure. It also offers a simplified capital and expenditure model for compute services as well as increased agility for cloud customers who can easily expand and contract their IT services as business needs change. As the most recent evolution in computing architecture, cloud computing is simply a further extension of the distributed computing model. Its key characteristics, such as multi-tenancy, massive scalability[2], elasticity, pay-per-use, and self-provisioned resources, are also those that may create new governance challenges for both cloud providers and their customers.

Manuscript received April 09, 2014.
Madesh Kumar S, PG Scholar, Dept. of CSE, HKBKCE, Bangalore.

A disguise must be invertible, secure and cheap[5].
– **Random Objects**. Create random numbers, vectors, matrices, functions, parameters, domains, etc.
– **Modify Linear Operators**. Lu = b becomes L(u+v) = b + c.
– **Modify Objects**. Add an object, multiply by an object.
– **Modify Domains**
Expand or restrict the problem domain.
Extend the interval of integration.
Solve a few linear equations.

Split a domain into two parts and use separate disguises on each. Change coordinates. There are several ways for A to attack the security [6]. There are many ways to disguise scientific problems. No single disguise technique applies to all problems. We have been able to disguise a wide variety of scientific computations [4], every one that we have tried. We believe there are computations which are **very hard** to disguise. The first goal of any outsourcing algorithm should be to hide as much information as possible about the actual computation from the helper [3], thus removing its ability to bias outputs or expose secrets. Obviously, software may also unintelligently fail. For example, the helper might contain a malicious bug that causes it to fail on every 1,000th invocation regardless of who is using it.

In various ways and forms, sequence comparisons arise in many applications other than molecular sequence comparison, notably, in text editing, speech recognition, machine vision, etc. In fact, the dynamic programming solution to this problem was independently discovered by no fewer than 14 different researchers [7] and is given a name by each discipline where it was independently discovered (Needleman-Wunsch by biologists, Wagner-Fischer by computer scientists, etc). When huge sequences are involved, the quadratic time complexity of the problem quickly becomes prohibitively expensive, requiring considerable power. Such supercomputing power is widely available, but

sending the data to such remote agents is problematic if the sequence data are sensitive, the outcome of the comparison is to be kept private, or both. In such cases[1], one can make a case for a technology that makes it possible for the customer to have the problem solved remotely but without revealing to the remote supercomputing sites either the inputs to the computation or its outcome.

Specifically, we first formulate private data owned by the customer for LP problem as a set of matrices and vectors. Linear programming is a mathematical technique for finding optimal solutions to problems that can be expressed using linear equations and inequalities. If a real-world problem can be represented accurately by the mathematical equations of a linear program, the method will find the best solution to the problem. Of course, few complex real-world problems can be expressed perfectly in terms of a set of linear functions[3]. Nevertheless, linear programs can provide reasonably realistic representations of many real-world problems — especially if a little creativity is applied in the mathematical formulation of the problem. [9], Linear programming is not a programming language like C++, Java, or Visual Basic. Linear programming can be defined as: "A mathematical method to allocate scarce resources to competing activities in an optimal manner when the problem can be expressed using a linear objective function and linear inequality constraints."[8] A linear program consists of a set of variables, a linear objective function indicating the contribution of each variable to the desired outcome, and a set of linear constraints describing the limits on the values of the variables. The "answer" to a linear program is a set of values for the problem variables that results in the best[1] — largest or smallest — value of the objective function and yet is consistent with all the constraints. [7]*Formulation* is the process of translating a real-world problem into a linear program. Once a problem has been formulated as a linear program, a computer program can be used to solve the problem. In this regard, solving a linear program is relatively easy. The hardest part about applying linear programming is formulating the problem and interpreting the solution[15].

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

II. PROBLEM STATEMENT

We consider a computation outsourcing architecture involving two different entities [1], 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),

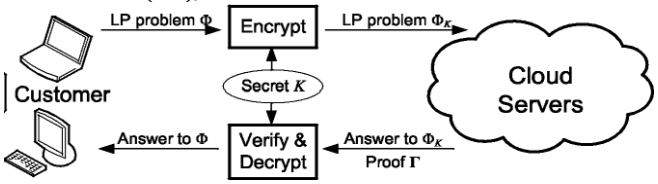

Fig. 1: Architecture of secure outsourcing linear programming problems in Cloud Computing

A related fear plagues cloud computing, where businesses buy computing time from a service, rather than purchasing, provisioning, and maintaining their own computing resources [1, 3].. A computation outsourcing scheme is a two-party protocol between a client C and server
S. The client chooses a computation task f and an input x. All of the equations and inequalities in a linear program must, by definition, be linear. A linear function has the following form:
a0 + a1 x1 + a2 x2 + a3 x3 + . . . + an xn = 0
In general, the *a*'s are called the *coefficients* of the equation; they are also sometimes called *parameters*. The important thing to know about the coefficients is that they are fixed values, based on the underlying nature of the problem being solved. Recently, Gaurha et al. [2] developed an efficient protocol for sequence comparisons in the secure two-party computation framework in which each party has a private string; the protocol enables two parties to compute the edit distance of two sequences such that neither party learns anything about the private sequence of the other party. and communication power, in our case the participant to whom all of the data and answer belong is asymmetrically weaker and is limited to a linear amount of computation and communication (hence cannot directly participate or help in each step of the quadratic-complexity dynamic programming solution).

III. THE PROPOSED SCHEMES

This section presents our LP outsourcing scheme for — not only the privacy protection of problem input/output which provides a *complete outsourcing* solution, but also its efficient result checking. Setty et al. [6] recently proposed a new public-key encryption system with which a mail server can perform keyword search on encrypted data with the help of the public-key holder. The protocol is too complicated. State-of-the-art PCP protocols partially address the concern about proof length, but they are intricate to the point of making a bug-free implementation difficult. Broadly speaking, outsourcing refers to a firm's external acquisition of inputs, services, or processes [3]. More specifically, some authors define outsourcing as an element of a firm's overall strategy or a firm's decision not to make a service or product internally, but to purchase it externally. The objective of a linear programming problem will be to maximize or to minimize some numerical value.
This value may be the expected net present value of a project or a forest property; or it may be the cost of a project; it could also be the amount of wood produced, the expected number of visitor-days at a park[4], the number of endangered species that will be saved, or the amount of a particular type of habitat to be maintained. One interesting corollary of our results is that mental poker can be played with any general public-key system. It differs from Shamir et. al's solution [5] in that we do not require the one-way functions used to be commutative, and that we can play it with a public-key system (instead of using private keys). (A solution with a special one-way function with publicized keys for playing mental poker was known in [2], but that solution depends on the special properties of the one-way function involved.) Moreover, the present solution uses much fewer bits as the number of cards becomes greater.

The **LP** problem does not necessarily have an optimal solution. There are three cases as follows.[3]

Normal: There is an optimal solution with finite objective value.

Infeasible: The constraints cannot be all satisfied at the same time.

Unbounded: For the standard form the objective function can be arbitrarily small while the constraints are all satisfied.

IV. SECURITY ANALYSIS

More specifically, a verifiable computation scheme consists of three phases:

Preprocessing A one-time stage in which the client computes some auxiliary (public and private) information associated with F.[2] This phase can take time comparable to computing the function from scratch, but it is performed only once, and its cost is amortized over all the future executions.

Input Preparation When the client wants the worker to compute $F(x)$, [9] it prepares some auxiliary (public and private) information about x. The public information is sent to the worker.

Output Computation and Verification Once the worker has the public information associated with F and x, it computes a string px which encodes the value $F(x)$ and returns it to the client. From the value px, [11] the client can compute the value $F(x)$ and verify its correctness.

Ciphertext-Policy FE: It is possible to obtain a ciphertext-policy FE scheme by constructing a dual of the above scheme. The structure of the ciphertext and key get interchanged. A key will encode a string w and a ciphertext[13] will encode an automaton M. Also, randomisation in Gp3 is done only for the key (i.e., components corresponding to the input string w). The same assumptions can also be used for the proof of security.

V. PERFORMANCE ANALYSIS

A. Theoretic Analysis

Fig 2: Classificatory framework.

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider [3].

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources [4] dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

B. Experiment Results

We now assess the practical efficiency of the proposed secure and verifiable LP outsourcing scheme with experiments. We implement the proposed mechanism including both the customer and the cloud side processes in Matlab and utilizes the MOSEK optimization [12] through its Matlab interface to solve the original LP problem (I) and encrypted LP problem K. In this way, the practical efficiency of the proposed mechanism can be assessed without a real cloud environment.

VI. RELATED WORK

A. Secret Voting.

Suppose a committee of m members wish to decide on a yes-no action. 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 Somindu et al. [9].

B. Oblivious Negotiation.

In principle, each one has a strategy of negotiation in mind. 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. [14] and many others.

C. Work on Delegating Computation and Cheating Detection

Detecting the unfaithful behaviors for computation outsourcing is not an easy task, even without consideration of input/output privacy[14]. 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.)

VII. CONCLUSION

In this paper, for the first time, we design which fulfills input/output privacy, cheating resilience, and efficiency. We formalize the problem of securely outsourcing LP computations in cloud computing, and provide such a practical mechanism. We also investigate classificatory

framework and derive a set of necessary and sufficient condition for result verification. Both security analysis and experiment results demonstrate the immediate practicality of the proposed mechanism. Such a resilience design can be bundled in the overall mechanism with close-to-zero additional overhead.

We plan to investigate some interesting future work as follows: 1) devise Action primarily consisting of activities directly aiming at producing plans; 2) A 'demonstration or pilot' aims to validate the technical and economic viability of a new or improved technology; 3) establish formal security framework; 4) extend our result to non-linear programming computation outsourcing in cloud.

REFERENCES

[1] M. J. Attalla and J. Li, "Secure outsourcing of sequence comparisons," Int. J. Inf. Sec., vol. 4, no. 4, pp. 277–287, 2005.
[2] N. Gaurha and Dr. Michael, "Data Security in Cloud Computing Using Linear Programming", International Journal of Emerging Technology and Advanced Engineering, Vol. 2, Issue 7, 2012.
[3] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in Proc. of TCC, pp. 264–282, 2005.
[4] C. Wang, K. Pen & J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," IEEE transactions on cloud computing , 2011.
[5] M Haungs, R Pandey, E Barr, and J. Barnes, "A fast connection-time redirection mechanism for internet application scalability", in *Proc. of STOC'87*, pp. 1–6, 1987.
[6] S Setty, R McPherson, J. Blumberg, and M Walfish, "Making Argument Systems for Outsourced Computation Practical", in *Proc. Of CRYPTO'10*, 2010.
[7] M Guirguis, A Bestavros, I Matta and Y Zhang, "Reduction of Quality (RoQ) Attacks on Internet End-Systems", in *Proc. of ASIACCS*, pp. 48–59, 2010.
[8] K Hashizume, D Rosado, E Medina and E B Fernandez, "An analysis of security issues for cloud computing", Journal of Internet Services and Applications 2013, 4:5
[9] Ramanna, "DFA-Based Functional Encryption: Adaptive Security from Dual System Encryption", in *Proc. of ICDCS'10*, 2010.
[10] Jiangtao & Mikhail," Secure and private collaborative linear programming", IEEE 2006.
[11] Andrew , "Protocols for Secure Computations", IEEE, 2007.
[12] M Shrivastava, "Security in Cloud Computing Using Linear Programming" , *International Journal Of Emerging Technology And Advanced Engineering (IJETAE)*, 2012.
[13] B Martini and R Choo, "An integrated conceptual digital forensic framework for cloud computing", Elsevier Ltd, 2012.
[14] S Qaisar and K Khawaja, "Cloud Computing: Network/Security Threats And Counter measures", Interdisciplinary Journal Of Contemporary Research In Business, Vol. 3, 2012.
[15] L Marchegiani, Enzo and Peruffo, "Revitalising the Outsourcing Discourse within the Boundaries of Firms Debate", Business Systems Review, Vol. 1, 2012.

Madesh Kumar S, is pursuing M.tech in Computer Science and Engineering at HKBK College of Engineering, Bangalore, Karnataka. His area of interest Cloud computing and Computer Networking.