International Journal of Advanced Computer and Mathematical Sciences
ISSN 2230-9624. Vol 3, Issue 4, 2012, pp 394-404
http://bipublication.com

PUBLIC LINEAR PROGRAMMING SOLUTION FOR THE DESIGN OF SECURE AND EFFICIENT COMPUTING IN CLOUD

K. Haripriya, M. Bhasker Rao, B. Ravi Raju

Dept of CSE, Bandari Srinivas Institute of Technology, Hyderabad, A.P, India

[Received-09/09/2012, Accepted-01/10/2012]

ABSTRACT:

This next generation of computing holds enormous potential to stimulate economic growth and enable governments to reduce costs, increase transparency and expand services to citizens. Cloud computing robust computational power to the society at reduced cost and 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. Despite the tremendous benefits, security is the primary obstacle that prevents the wide adoption of this promising computing model, especially for customers when their confidential data are consumed and produced during the computation.

Keywords: Linear Programming, computing and optimization tasks, sensitive information, Privacy preserving

I. EXISTING SYSTEM

Cloud computing builds off a foundation of technologies such as grid computing, which includes clustering, server virtualization and dynamic provisioning, as well as SOA shared services and large-scale management automation. For the better part of a decade, Oracle has been the leader in these areas with thousands of customer successes and high level of investment. Today, Oracle offers the industry's most complete, open and integrated products and services to enable public, private and hybrid clouds. [31]

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

Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customers' direct control over the systems that consume and produce their data during the computation, which inevitably brings in new security concerns and challenges towards this promising

computing model [2]. On the one hand, the outsourced computation workloads often contain sensitive information, such as the business financial records, proprietary research data, or personally identifiable health information etc.

To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing [2] so as to provide end to- end data confidentiality assurance in the cloud and beyond. However, ordinary data encryption techniques in essence prevent cloud from performing any meaningful operation of the underlying plaintext data [3], making the computation over encrypted data a very hard problem. On the other hand, the operational details inside the cloud are not transparent enough to customers [4]. 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. 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. In short, practically efficient mechanisms with immediate practices for secure computation outsourcing in cloud are still missing.

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

Five Essential Characteristics

 On-demand self service –Users are able to provision, monitor and manage computing resources as needed without the help of human administrators

 Broad network access – Computing services are delivered over standard networks and heterogeneous devices

 Rapid elasticity – IT resources are able to scale out and in quickly and on an as needed basis

 Resource pooling – IT resources are shared across multiple applications and tenants in a non-dedicated manner

 Measured service – IT resource utilization is tracked for each application and tenant, typically for public cloud billing or private cloud chargeback

Three Service Models

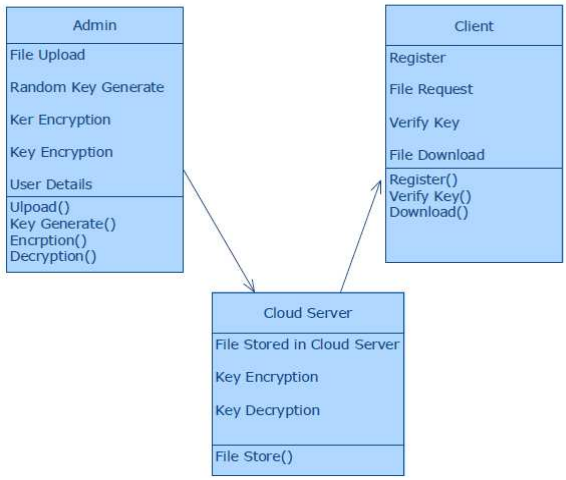
 Software as a Service (SaaS) – Applications delivered as a service to end-users typically through a

Web browser. There are hundreds of SaaS service offerings available today, ranging from horizontal enterprise applications to specialized applications for specific industries, and also consumer applications such as Web-based email. *Oracle CRM On Demand* is an example of a SaaS offering that provides both multi-tenant as well as single-tenant options, depending on the customer's preference. Oracle also offers enterprise-grade enabling technology to Independent Software Vendors (ISVs) to build their own SaaS offerings. Oracle calls this enabling technology the *Oracle Platform for SaaS*. Hundreds of ISVs have built their SaaS offering on top of the Oracle Platform for SaaS.

 Platform as a Service (PaaS) – An application development and deployment platform delivered as a service to developers who use the platform to build, deploy and manage SaaS applications. The platform typically includes databases, middleware and development tools, all delivered as a service via the Internet. PaaS offerings are often specific to a programming language or APIs, such as Java or Python. A virtualized and clustered grid computing architecture is often the basis for PaaS offerings, because grid provides the necessary elastic scalability and resource pooling. Oracle offers a comprehensive PaaS product offering for public cloud service providers as well as enterprise customers to build their own public clouds. Oracle calls this the *Oracle PaaS Platform* (more on this later in this paper).

 Infrastructure as a Service (IaaS) – Compute servers, storage, and networking hardware delivered as a service. This infrastructure hardware is often virtualized, so virtualization, management and operating system software are also part of IaaS as well. An example of IaaS is Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3). Oracle does not offer IaaS cloud services, but Oracle provides hardware and software products to other IaaS providers to enable their public cloud services, and also offers the same technologies to enterprises for private use. [31]

Class Diagram:



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

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 hones 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".

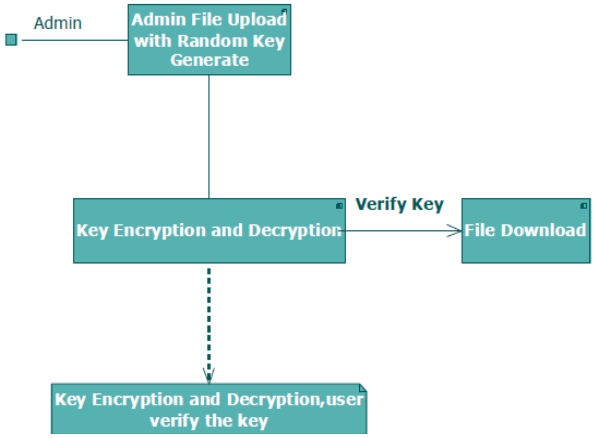
.

II. PROPOSED SYSTEM

IMPLEMENTATION

Despite the tremendous benefits, outsourcing computation to the commercial public cloud is also depriving customers' direct control over the systems that consume and produce their data during the computation, which inevitably brings in new security concerns and challenges towards this promising computing model. 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.

Component Diagram:



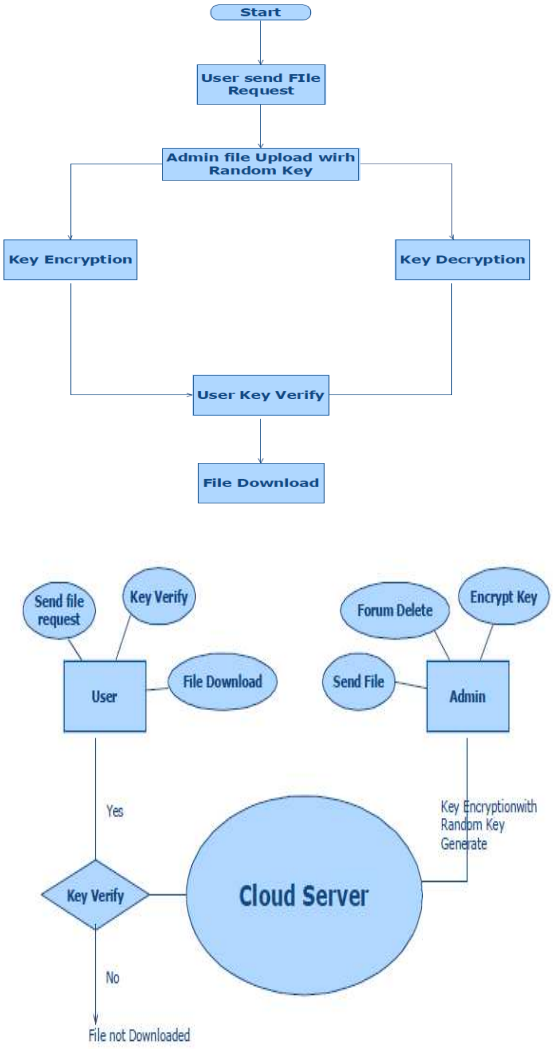
To combat against unauthorized information leakage, sensitive data have to be encrypted before outsourcing. so as to provide end to- end data confidentiality assurance in the cloud and beyond. However, ordinary data encryption techniques in essence prevent cloud from performing any meaningful operation of the underlying plaintext data, 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. 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 hones 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"

Dataflow diagram



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 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, 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. 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.

Fully homomorphic encryption (FHE) scheme, a general result of secure computation outsourcing has been shown viable in theory, where the computation is represented by an encrypted combinational Boolean circuit that allows to be evaluated with encrypted private inputs.

Module Description:

1. **Mechanism Design Framework**
2. **Basic Techniques**
3. **Enhanced Techniques via Affine Mapping**
4. **Result Verification**

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.

• KeyGen(1k) → {K}. *This is a randomized key generation algorithm which takes a system security parameter* k, *and returns a secret key* K *that is used later by customer to encrypt the target LP problem.*

• ProbEnc(K,_) → {_K}. *This algorithm encrypts the input tuple* _ *into* _K *with the secret key* K. *According to problem transformation, the encrypted input* _K *has the same form as* _, *and thus defines the problem to be solved in the cloud.*

• ProofGen(_K) → {(y, □)}. *This algorithm augments a generic solver that solves the problem* _K *to produce both the output* y *and a proof* □. *The output* y *later*

decrypts to x, *and* □ *is used later by the customer to verify the correctness of* y *or* x.

• ResultDec(K,_, y, □) → {x,⊥}. *This algorithm may choose to verify either* y *or* x *via the proof* ▪ . *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.*

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 (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 ⇒ A′x = b′

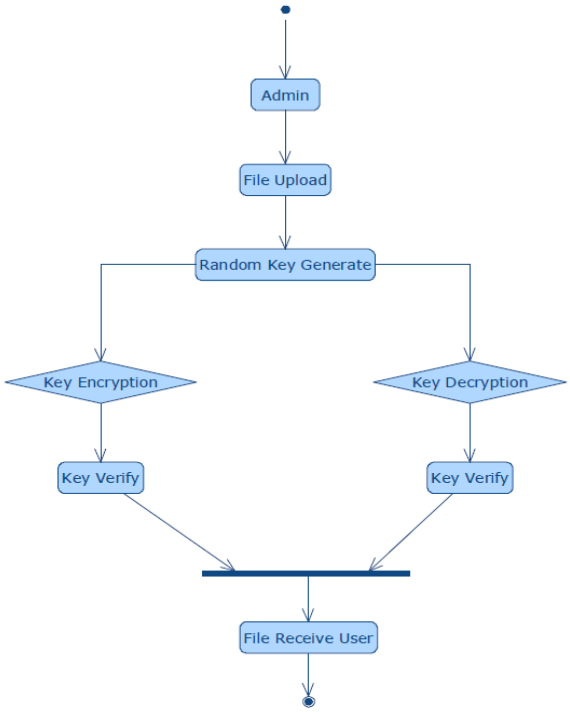
where A′ = QA and b′ = Qb.

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 _ 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 _ 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.

Activity diagram



Result Verification

Till now, we have been assuming the server is honestly performing the computation, while being interested learning information of original LP problem. However, such semihonest model is not strong enough to capture the adversary behaviors in the real world. In many cases, especially when the computation on the cloud requires a huge amount of computing resources, there exists strong financial incentives for the cloud server to be “lazy”. They might either be not willing to commit service-level-agreed computing resources to save cost, or even be malicious just to sabotage any following up computation at the customers. Since the cloud server promises to solve the LP problem _K = (A′,B′, b′, c′), we propose to solve the result verification problem by

designing a method to verify the correctness of the solution y of _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.

• *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 in Eq. (1), the objective function can be arbitrarily small while the constraints are all satisfied.

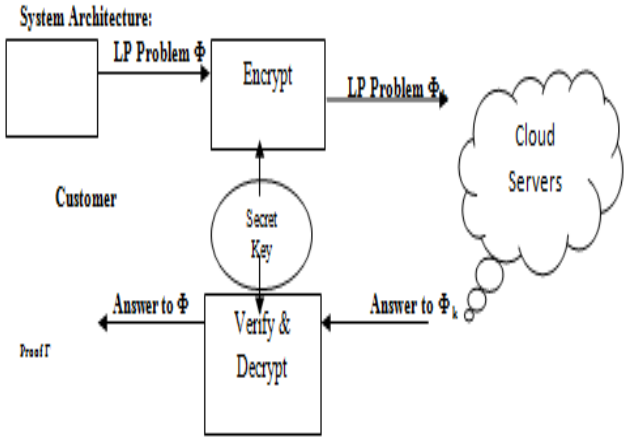


Fig: Architecture of Secured Outsourcing liner Programming problems in Cloud Computing

Algorithm Used:

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.

ProbEnc(K, Φ) → {Φ_K}.

This algorithm encrypts the input tuple _ into _K with the secret key K. According to problem transformation, the encrypted input _K has the same form as _, 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 _K to produce

both the output y and a proof □. The output y later decrypts to x, and □ 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 □. 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.

. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual

software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

3 Acceptance Testing

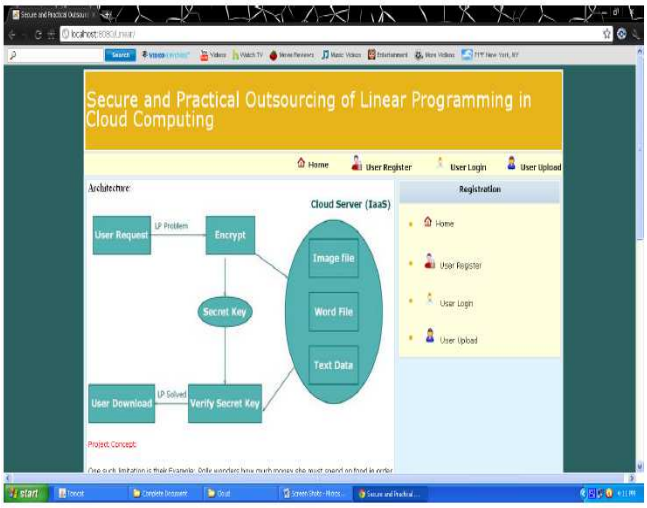
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

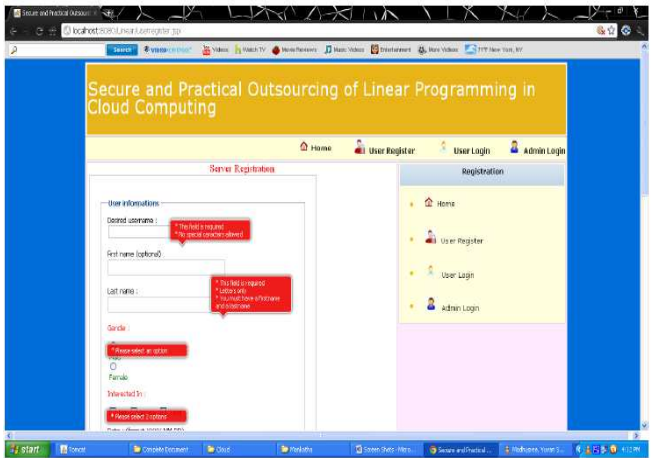
III CONCLUSIONS

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.

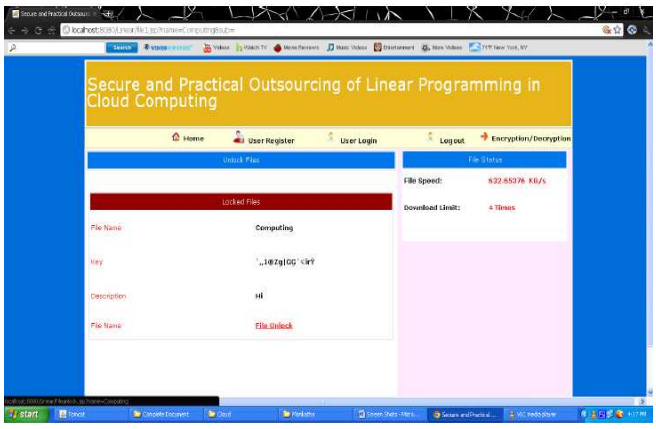
Home page snap shot



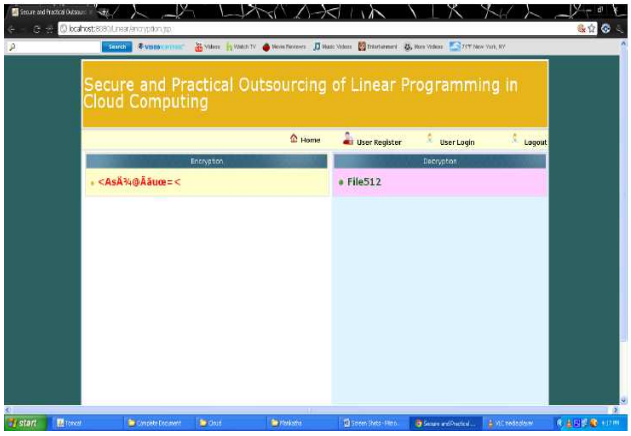
User registrataion



Locked files



Encryption/decryption



We plan to investigate some interesting future work as follows: 1) devise robust algorithms to achieve numerical stability; 2) explore the sparsity structure of problem for further efficiency improvement; 3) establish formal security framework; 4) extend our result to non-linear programming computation outsourcing in cloud.

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 CRYPTO'10*, 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'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," *Numer. Math.*, 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, 1979.

[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. IEEE TRANSACTIONS ON CLOUD COMPUTING April 10-15, 2011
[30] Secure and Practical Outsourcing of Linear Programming in Cloud Computing Cong Wang, Kui Ren, and Jia Wang Department of Electrical and Computer Engineering Illinois Institute of Technology, Chicago, IL 60616, USA Email: {cong, kren, jwang}@ece.iit.edu
[31] Oracle Cloud Computing, An Oracle White Paper, May 2010, http://www.oracle.com/us/technologies/cloud/oracle-cloud-computing-wp-076373.pdf