ISSN:0975-9646

Secured Auditing of Outsourced Data using Linear Programming in Cloud

[1] Satheesh Kavuri,[2] Dr. Gangadhara Rao Kancharla, [1]Sowmya Koneru,[1] Sandeep Kotte

[1]*Department of Computer Science & Engineering*
Dhanekula Institute of Engineering & Technology, Vijayawada
[2]*Department of Computer Science & Engineering*
Acharya Nagarjuna University, Guntur

Abstract-Cloud computing can be seen as an innovation in different ways. From a technological perspective it is an advancement of computing, which's history can be traced back to the construction of the calculating machine. Cloud Computing has great potential of providing robust computational power to the society at reduced cost. It 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. 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 the computation result. In order to achieve practical efficiency, our mechanism design explicitly decomposes the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The resulting flexibility allows us to explore appropriate security/efficiency tradeoff via higher-level abstraction of LP computations than the general circuit representation. In particular, by formulating private data owned by the customer for LP problem as a set of matrices and vectors, we are able to develop a set of efficient privacy-preserving problem transformation techniques, which allow customers to transform original LP problem into some arbitrary one while protecting sensitive input/output information. To validate the computation 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.

I. INTRODUCTION

The term cloud computing is sometimes used to refer to a new paradigm – some authors even speak of a new technology – that flexibly offers IT resources and services over the Internet. 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. 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 hardware and software and/or the operational overhead there in. 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. 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. 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. 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. 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, or multi-round interactive protocol executions, or huge communication complexities, are involved (detailed discussions in Section VI). In short, practically efficient mechanisms with immediate practices for secure computation outsourcing in cloud are still missing. we propose to explicitly decompose the LP computation outsourcing into public LP solvers running on the cloud and private LP parameters owned by the customer. The flexibility of such a decomposition allows us to explore higher-level abstraction of LP computations than the general circuit representation for the practical efficiency. Specifically, we first formulate private data owned by the customer for LP problem as a set of matrices and vectors. This higher level representation allows us to apply a set of efficient privacy-preserving problem transformation techniques, including matrix multiplication and affine mapping, to transform the original LP problem into some 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, 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.

II. THE LAYERS OF CLOUD COMPUTING

Cloud computing is based on a set of many pre-existing and well researched concepts such as distributed and grid computing, virtualization or Software-as-a-Service. Although, many of the concepts don't appear to be new, the real innovation of cloud computing lies in the way it provides computing services to the customer. Various business models have evolved in recent times to provide services on different levels of abstraction. These services include providing software applications, programming platforms, data-storage or computing infrastructure. Classifying cloud computing services along different layers is common practice in the industry23. Wang et al. for example describe three complementary services, Hardware-as-a- Service (HaaS), Software-as-a-Service (SaaS) and Data-as-a-Service (DaaS). These services together form Platform-as-a-Service (PaaS), which is offered as cloud computing24. In an attempt to obtain a comprehensive understanding of cloud computing and its relevant components, Youseff, Butrico and Da Silva were among the first who suggested a unified ontology of cloud computing25. According to their layered model, cloud computing systems fall into one of the following five layers: applications, software environments, software infrastructure, software kernel, and hardware. Each layer represents a level of abstraction, hiding the user from all underlying components and thus providing simplified access to the resources or functionality. In the following section we are going to describe each layer of Youseff's Butrico's and Da Silva's model.

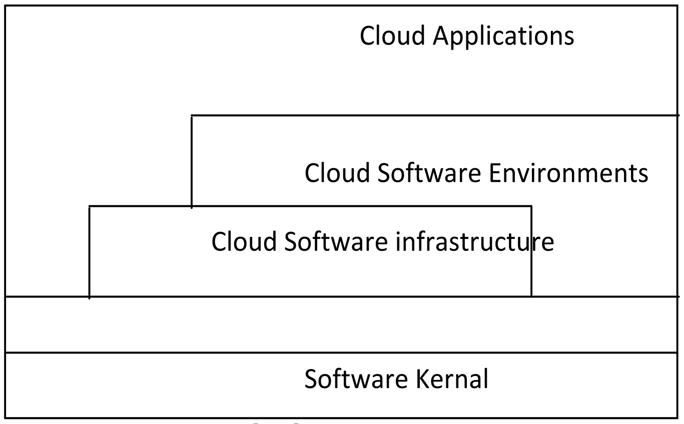

Figure1. Cloud Computing Structure

Cloud Application Layer

When it comes to user interaction, the cloud application layer is the most visible layer to the end-customer. It is usually accessed through web-portals and thus builds the front-end, the user interacts with when using cloud services. A Service in the application layer may consist of a mesh of various other cloud services, but appears as a single service to the end-customer. This model of software provision, normally also referred to as *Software-as-a-Service*, appears to be attractive for many users. Reasons for this are the reduction of software and system maintenance costs, the shift of computational work from local systems into the cloud, or a reduction of upfront investments into hardware and software licenses. Also the service provider has advantages over traditional software licensing models. The effort for software upgrades is reduced, since patches and features can be deployed centrally in shorter cycles. Depending on the pricing model a continuous revenue stream can be obtained. However, security and availability are issues that still need to be addressed. Also the migration of user data is a task that should not be underestimated. Examples for applications in this layer are numerous, but the most prominent might be Sales force's Customer Relationships Management (CRM), which include word-processing, spreadsheet and calendaring.

Cloud Software Environment Layer

The cloud software environment layer (also called software platform layer) provides a programming language environment for developers of cloud applications. The software environment also offers a set of well-defined application programming interfaces (API) to utilize cloud services and interact with other cloud applications. Thus developers benefit from features like automatic scaling and load balancing, authentication services, communication services or graphical user interface (GUI) components. However, as long as there is no common standard for cloud application development, lock-in effects arise, making the developer dependent on the proprietary software environment of the cloud platform provider. This service, provided in the software environment layer is also referred to as *Platform-as-a-Service*. A known example of a cloud software platform is Google's App Engine29, which provides developers a phyton runtime environment and specified APIs to develop applications for Google's cloud environment. Another example is Sales force's Apex change platform30 that allows developers to extend the Sales force CRM solution or even develop entire new applications that runs on their cloud environment. As we will highlight in Chapter 4.1 one can also look at the cloud platform from a value network or business model perspective. In that sense, the cloud platform can act as a market place for applications.

Cloud Software Infrastructure Layer

The cloud software infrastructure layer provides resources to other higher-level layers, which are utilized by cloud applications and cloud software platforms. The services offered in this layer are commonly differentiated into computational resources, data storage, and communication. Computational resources in this context are usually referred to as *Infrastructure-as-a-Service (IaaS)*. Virtual Machines are the common form of providing computational resources to users, which they can fully administrate and configure to fit their specific needs. Virtualization technologies can be seen as the enabling technology for IaaS, allowing data center providers to adjust resources on demand, thus utilizing their hardware more efficiently. The downside of the medal is the lack of a strict performance allocation on shared hardware resources. Due to this, infrastructure providers cannot give strong performance guarantees which result in unsatisfactory service level agreements (SLA). These weak SLAs propagate upwards in the cloud stack, possibly leading to availability problems of cloud applications. The most prominent examples of IaaS are Amazon's Elastic Compute Cloud31 and Enomalism's Elastic Computing Infrastructure32. There are also some academic open source projects like Eucalyptus33 and Nimbus34. In analogy to computational resources data storage within the cloud computing model is offered as *Storage-as-a-Service*. This allows users to obtain demand-flexible storage on remote disks which they can access from everywhere. Like for other storage systems, tradeoffs must be made between the partly conflicting requirements: high availability, reliability, performance, replication and data consistency, which in turn are manifested in the service providers SLAs. Examples of Storage-as-a-Service are Amazon's Elastic Block Storage (EBS)35 or its Simple Storage Service (S3)36 and Rackspace's Cloud Files37. In addition, to simple storage space, data can be offered as service as well. Amazon for example offers the human genome or the US census as public data sets to use for other services or analytics38. A fairly new idea is *Communication-as-a-Service (CaaS)*, which shall provide quality of service ensured communication capabilities such as network security, dedicated bandwidth or network monitoring. Audio and video conferencing is just one example of cloud applications that would benefit from CaaS. So far this service is only a research interest rather than in commercial use. However, Microsoft's Connected Service Framework (CSF)39 can be counted into this class of services. As Figure 1 shows, cloud applications must not necessarily be developed upon a cloud software platform, but can also run directly on the cloud software infrastructure layer or even the software kernel, thus bypassing the aforementioned layers. Although this approach might offer some performance advantages, it is directly dependent on lower level components and does not make use of development aids such as the automatic scaling provided by the cloud software platform.

Software Kernel Layer

The software kernel layer represents the software management environment for the physical servers in the datacenters. These software kernels are usually implemented as operation system kernel, hypervisor, virtual machine monitor or clustering middleware. Typically, this layer is also the level where grid computing applications are deployed. Globus40 is an example of a successful grid middleware. At this layer, cloud computing can benefit from the research already undertaken in the grid computing research community.

Hardware / Firmware Layer

At the bottom end of the layered model of cloud computing is the actual physical hardware, which forms the backbone of any cloud computing service offering. Hardware can also be subleased from datacenter providers to, normally, large enterprises. This is typically offered in traditional outsourcing plans, but in a as-a-service context also referred to as *Hardware-asa- Service (HaaS)*.

III. IT AUDITING USING CONTROLS TO PROTECT INFORMATION

Software as a Service (SaaS) In this model, you will access the cloud provider's applications, which are running on a cloud infrastructure. The applications are accessible from client devices through a thin client interface such as a web browser (for example, web-based e-mail). As the consumer, you don't manage or control the data center, network, servers, operating systems, middleware, database management system (DBMS), or even individual application capabilities (with the possible exception of limited user-specific application configuration settings), but you do have control over your data.

Platform as a Service (PaaS) In this model, you will deploy applications you created or acquired onto the provider's cloud infrastructure, using programming languages and tools supported by the cloud provider. As the consumer, you don't manage or control the data center, network, servers, operating systems, middleware, or DBMS, but you do have control over your data and the deployed applications and possibly application hosting environment configurations.

Infrastructure as a Service (IaaS) In this model, processing, storage, networks, and other fundamental computing resources are rented from the cloud provider. This allows you to deploy and run arbitrary software, which can include operating systems and applications. As the consumer, you don't manage or control the data center or network, but you do have control over your data and the operating systems, middleware, DBMS, and deployed applications.

IV. PROBLEM STATEMENT

the data be encrypted to protect against possible compromise. This reduces the risk of a breach impacting the confidentiality or integrity of your data. If you have unencrypted data in a shared environment (such as cloud computing), you can assume that it is no longer confidential. This section presents our LP outsourcing scheme which provides a *complete outsourcing* solution for – not only the privacy protection of problem input/output, but also its efficient result checking. 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. We then discuss effective result verification by leveraging the duality property of LP. Finally, we give the full scheme description.

Linear Programming

An optimization problem is usually formulated as a mathematical programming problem that seeks the values for a set of decision variables to minimize (or maximize) an objective function representing the cost subject to a set of constraints. For linear programming, the objective function is an affine function of the decision variables, and the constraints are a system of linear equations and inequalities. Since a constraint in the form of a linear inequality can be expressed as a linear equation by introducing a non-negative slack variable, and a free decision variable can be expressed as the difference of two non-negative auxiliary variables, any linear programming problem can be expressed in the following standard form, minimize cT x subject to Ax = b, x ≥ 0. 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. In this paper, we study a more general form as follows, Minimize cT x subject to Ax = b, Bx $\geq$ 0. In Eq. (2), we replace the non-negative requirements in Eq. by requiring that each component of Bx to be nonnegative, where B is an n × n non-singular matrix, i.e. Eq. (2) de- generates to Eq. (1) when B is the identity matrix. Thus, the LP problem can be defined via the tuple Φ = (A, B, b, c) as input, and the solution x as output.

V. RESULT ENHANCEMENT

The 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(n2). We now assess the practical efficiency of the proposed secure and verifiable LP outsourcing scheme with experiments. We implement the proposed mechanism including both the customer and the cloud side processes in Matlab and utilize the MOSEK optimization [20] through its Matlab interface to solve the original LP problem Φ and encrypted LP problem ΦK . 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. We also ignore the communication latency between the customers and the cloud for this application since the computation dominates the running time as evidenced by our experiments. 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 experimental 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, 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 cloud and the time for the customer . Note that since each KeyGen would generate a different key, the encrypted LP problem ΦK generated by ProbEnc would be different and thus result in a different running time to solve it. The cloud and customer reported in Table I are thus the average of multiple trials. We propose to assess the practical efficiency by two characteristics calculated from original , cloud, and customer. The *Asymmetric Speedup*, calculated as original, 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 cloud, represents the overhead introduced to the overall computation by the proposed mechanism.

VI. CONCLUSION

In this paper the historic development of providing IT resources, cloud computing has been established as the most recent and most flexible delivery model of supplying information technology. It can be seen as the consequent evolution of the traditional on-premise computing spanning outsourcing stages from total to the selective, and from the multi-vendor outsourcing to an asset-free delivery. While from a technical perspective, cloud computing seems to pose manageable challenges, it rather incorporates a number of challenges on a business level, both from an operational as well as from a strategic point of view. As laid out above, cloud computing in its current stage also holds a number of contributions for both theory and practice that this article could reveal and that will be addressed. 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 demonstrates the immediate practicality of the proposed mechanism.

REFERENCES

[1] A. C.-C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. of FOCS'82*, 1982, pp. 160–164.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc of STOC*, 2009, pp. 169–178.

[3] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. Springer, 2008.

[4] 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.

[5] 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.

[6] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT press, 2008.

[8] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, vol. 13, pp. 354–356, 1969.

[9] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. of STOC'87*, 1987, pp. 1–6.

[10] MOSEK ApS, "The MOSEK Optimization Software," Online at http: //www.mosek.com/, 2010.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. of EUROCRYPT'99*, 1999, pp. 223–238.

[12] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[13] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

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

[15] 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.

[16] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Proc. of CollaborateCom*, Nov. 2006.

[17] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: interactive proofs for muggles," in *Proc. of STOC*, 2008, pp. 113–122.

[18] Sun Microsystems, Inc., "Building customer trust in cloud computing with transparent security," 2009, online at https://www.sun.com/offers/details/sun transparency.xml.

[19] 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.

[20] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in Proc. of TCC, 2005, pp. 264–282.

[21] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," Int. J. Inf. Sec., vol. 4, no. 4, pp. 277–287, 2005.

[22] 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.