Privacy Assured Delegation of Massive Linear Programming Computational Workload

Ms. Dhanya G. A
Dept of Computer Science and Engineering,
Channabasaveshwara Institute of Technology,
Gubbi (T), Tumkur (D), Karnataka, India

Mrs. Jyothi K. S
Dept of Computer Science and Engineering,
Channabasaveshwara Institute of Technology,
Gubbi (T), Tumkur (D), Karnataka, India

***Abstract*— Cloud Computing is a membership based service where you can obtain the networked storage space and computer resources. In this cloud computing model, the customers connect to the cloud to access IT resources which are charged and provided on demand services. This model is composed of five important characteristics, three service models and four deployment models. Users can store their data in the cloud and there is a lot of personal information and secure data that people store on their computers, and this information is now being transferred to the cloud. So we must ensure the security of user's data, which is stored in the cloud. In this paper we present privacy assured delegation mechanism for linear programming computations in the cloud computing environment. Linear programming is a computational tool, which is used to analyze and optimize real world systems. Here we built the privacy assured LP delegation mechanism using a different approach i.e. iterative method which is easy to implement practically and requires only simple matrix-vector operations. In this mechanism customer can keep confidential both input and output of the computation secure by using additive homomorphic encryption method and can use the cloud for iteratively finding successive approximations to the LP solution. For untrusted cloud cheating detection, we use efficient verification mechanism that allows customers to verify all results from cloud effectively.**

Keywords- privacy, linear programming, additive homomorphic encryption, cloud computing, delegation

I. INTRODUCTION

Cloud computing is a model that provides easy to use, on-demand network access to a shared pool of computing resources that can be quickly provided and released with minimum management effort or service provider interaction. It has large potential of providing computational power at reduced cost and allows the customers with limited computational resources to delegate their massive computation workloads to the cloud, and thus customers can enjoy the massive computational power, bandwidth, storage, and even appropriate software that can be shared in a pay-per-use manner [1]. An example of cloud computing architecture is shown in fig 1. Even though with these immense benefits, security is the main obstacle that prevents the wide adoption of this computing model, especially for the customers when their confidential data are consumed and produced during the computation. Another reason is that the delegated computation workloads contain confidential information. So in order to avoid unauthorized information leakage, this data have to be encrypted before delegating it to the cloud. If we apply ordinary data encryption techniques this restricts the cloud from performing any useful operation of the given plaintext data and makes the computation of encrypted data very difficult [2]. And also the operational details inside the cloud are not visible enough to customers [3]. As a result cloud server may return incorrect results. Along with this some software bugs, hardware failures, and also outsider attacks may also affect the quality of the computed results. So, 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 computed result, it would be hard to expect cloud customers to turn over control of their workloads from local machines to cloud only based on its economic savings and resource flexibility. For practical consideration, such a design should further ensure that customers perform fewer amounts of operations than completing the computations by themselves directly.

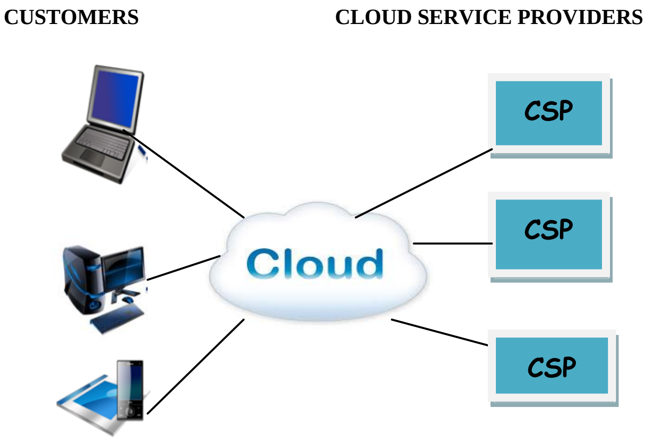

Fig 1: Cloud Computing Architecture Example

Here we design secure mechanism of delegating LP computation by using a different method i.e. iterative method, in this method the solution is extracted by finding successive approximations to the solution until the required accuracy is obtained. Comparing it to direct method, iterative method only requires simple matrix-vector operations with $O(n^2)$ cost, which is very easy to implement practically. Here we use the additive homomorphic encryption scheme, e.g., the Paillier cryptosystem [4] that allows computational customers to securely utilize the cloud for finding successive approximations to the computed solution in a privacy-preserving and cheating-resilient manner. For a linear system with n×n coefficient matrix, the proposed mechanism is based on a one-time amortizable setup with O(n^2) cost. Then, in each iterative algorithm execution, our proposed mechanism only requires O(n) local computational burden to the customer and demands no impractical memory i.e. eliminates the expensive IO cost. To verify computation result honesty, we also propose a very efficient cheating detection mechanism to effectively verify in one batch of all the computation results by the cloud server from previous algorithm iterations with high probability.

1) We propose to solve the problem of privacy assured delegation of massive LP computational workload using iterative methods, and provide mechanism designs which fulfill input/output privacy, cheating resilience, and efficiency.
2) Within each iteration, it requires only O (n) computation burden i.e. less burden for the customer and demands no unrealistic IO cost, as a result our mechanism brings computational savings, in terms of both time and memory requirements [5].
3) To design a efficient batch verification mechanism we use the algebraic property of matrix-vector operations, which allows customers to verify all results of previous iterations from cloud effectively.

II. PROBLEM STATEMENT

A. Threat Model

Here we consider a architecture of secure computation delegation which involves cloud customer and cloud server as shown in Fig. 2. The customer who has a massive LP problem to be solved which is denoted as Φ. But due to the lack of computing resources, he cannot perform such expensive computation locally. So, the customer sends it to cloud server for solving the LP problem. In order to protect the data, the customer first uses a secret key K to transform Φ into some encrypted form Φk. After that, based on encrypted Φk, the customer starts the computation delegation protocol with CS, and uses the cloud resources in a privacy-preserving manner. The CS helps the customer to find the answer of Φk, but is supposed to learn as little as possible on the confidential information in Φ. And after receiving the solution of encrypted problem Φk, the customer should be able to verify the answer first. If that answer is correct, then he then uses the secret K to map the output into the desired answer for the original problem Φ.

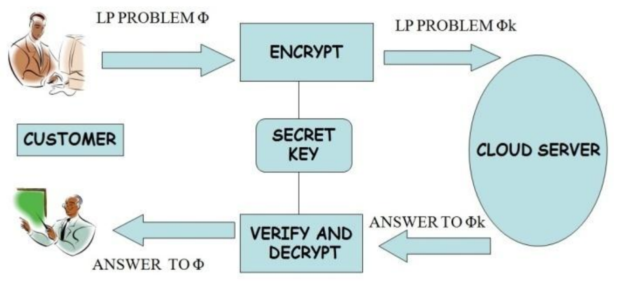

Fig 2 Architecture of secure outsourcing large-scale LP computations in cloud computing

The problem with this existing approach is that for encrypting the data if we use ordinary encryption techniques then computation over encrypted data becomes very difficult. The security problem faced by the computation model is that comes from the harmful behavior of CS, [6]. The CS may be very much interested in studying the encrypted input which was sent by the customer and the encrypted output produced by the computation in order to learn the confidential information. In addition to being interested in knowing the sensitive input/output information of Φ, CS can also behave unfaithfully or intentionally disrupt the computation, i.e. to lie about the result in order to save the computing resources, and hopes not to be caught.

B. Design Goals

1) Input/output confidentiality: Confidential information from the customer's private data cannot be obtained by the cloud server when performing the LP (linear programming) computation;
2) Vigorous cheating detection: Computation output from cloud server must be verified successfully by the customer. No output from cheating cloud server can pass the verification.
3) Efficiency: The local LP computing burden, for the customer should be much less than solving the original LP on his own.

III. RELATED WORK

In the literature several protocols for cryptography has been proposed for solving linear programming problems by secure multiparty computations[7][8], these protocols are not suitable for solving large size problems and do not address the irregularity among computation power of cloud and the customer, hence involve computation burden. The work developed under SMC model focus on traditional direct method [9][11],for computing linear equations and work well for small size problems. Lastly they do not consider result verification as a serious security requirement [10]. Even though result verification is done they allow server to see the data and result it is computing[12]. So an efficient and secure LP computational data outsourcing mechanism is required.

IV. PROPOSED METHODOLOGY

A. Iterative Method

Here we are using iterative method for problem solving due to its ease of implementation and relatively less computational power consumption, including the memory and storage IO requirement [5]. A system of linear equations is written as

$$Ax = b \qquad (1)$$

Where x is the n × 1 vector of unknowns, A is an n × n (nonsingular) coefficient matrix, b is an n × 1 right-hand side vector. We use Jacobi iteration [18] and start with the decomposition: A =D + R, Where D is the diagonal component, R is the remaining matrix. Then, the (1) can be written as Ax = (D+R) x=b, and reorganized as: x = -D^{-1}. R. x + D^{-1}. b. If we denote iteration matrix T = -D^{-1} . R and c = D^{-1} . b, the above iterative equations can be represented as

$$x^{(k+1)} = T \cdot x^{(k)} + c \qquad (2)$$

B. Additive Homomorphic Encryption

In our methodology we use a semantically secure encryption scheme with additive homomorphic property. Given two integers x_1 and x_2, we have $Enc(x_1+x_2) = Enc(x_1) * Enc(x_2)$, and also $Enc(x_1*x_2) = Enc(x_1)^{x}{}_2$. In our implementation we adopt the Paillier cryptosystem [4]. For vector $x = (x_1, x_2, \ldots, x_n)^T \in (Z_N)^n$, we use Enc(x) to denote the coordinate-wise encryption of x: $Enc(x) = (Enc(x_1), Enc(x_2), \ldots\ldots, Enc(x_n))T$ For some $n \times n$ matrix T each of the component T[i, j] in T is from Z_N, we denote the component-wise encryption of T as Enc(T), and we have Enc(T[i, j]) = Enc(T[i, j]).

V. MODULES DESIGN FRAMEWORK

Our proposed method consists of three phases Problem Transformation, Problem Solving and Result Verification, and the system flow diagram is shown in the figure 3.

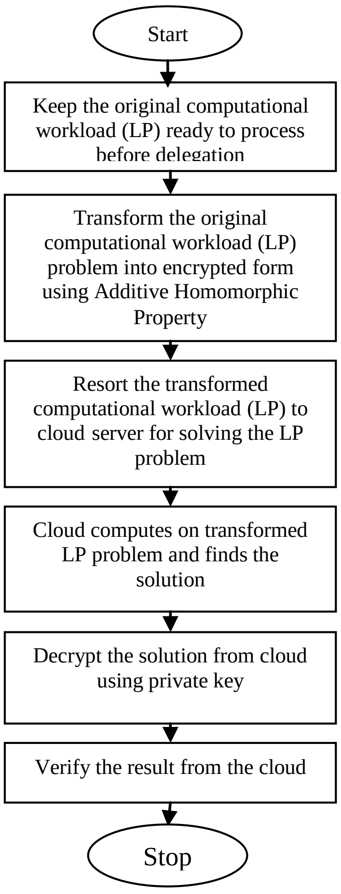
Start
Keep the original computational workload (LP) ready to process before delegation
Transform the original computational workload (LP) problem into encrypted form using Additive Homomorphic Property
Resort the transformed computational workload (LP) to cloud server for solving the LP problem
Cloud computes on transformed LP problem and finds the solution
Decrypt the solution from cloud using private key
Verify the result from the cloud
Stop

Fig 3 System flow diagram

A Problem Transformation by Encrypting

In this phase, the cloud customer uses a randomized key generation algorithm and transforms the LP problem into some encrypted form Φ_k. The customer picks a random vector $r \in IR^n$ as his secret key material, and rewrites Ax=b (1) as new LP problem.

$$Ay = b' \qquad (3)$$

Where $y = x + r$ and $b' = b + Ar$. Next, equation (3) can be rewritten as

$$y^{(k+1)} = T \cdot y^{(k)} + c' \qquad (4)$$

Where $T = -D^{-1} . R$, $c' = D^{-1} . b'$ and $A = D + R$. Now the problem input $\Phi=(A , b)$ is changed to tuple $\Phi_k= (T, c')$

The following algorithm shows the protocol execution for the phase of Problem Transformation.

```
Algorithm 1: Problem Transformation Phase
Data: original LP problem $\Phi = (\mathbf{A}, \mathbf{b})$
Result: transformed LP problem as shown in Eq. (4)
Begin
 1 pick random $\mathbf{r} \in R^n$ ;
 2 compute $\mathbf{b}' = \mathbf{b} + \mathbf{Ar}$, and $\mathbf{c}' = \mathbf{D}^{-1} \cdot \mathbf{b}'$;
 3 replace tuple $(\mathbf{x}, \mathbf{c})$ in Eq. (2) with $(\mathbf{y} = \mathbf{x} + \mathbf{r}, \mathbf{c}')$;
return transformed LP problem as Eq. (4);
```

B Problem Solving using Iterative method

In this phase, the cloud customer uses the encrypted form Φ_k of LP and starts the computation delegation process. In case of using the iterative methods, the protocol ends when the solution within the required accuracy is found. Our goal is to allow the customer securely utilize the cloud for the most expensive computation T. $y^{(k)}$ in (4) for each algorithm iteration, k =1, 2, . . . , L. we assume our protocol of solving LP works over integers. All arithmetic is modular with respect to the modulus N of the additive homomorphic encryption, and the modulus is large enough to contain the answer. For the first iteration, the customer starts initial guess on the vector $y^{(0)} = (y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)})^T$ and then sends it to the cloud. The cloud server, in possession of the encrypted matrix Enc(T), computes the value $Enc(T . y^{(0)})$ using the homomorphic property of the encryption:

$$Enc\left(T \cdot y^{(0)}\right)[i] = Enc\left(\sum_{j=1}^{n} T[i,j] \cdot y_j^{(0)}\right) = \prod_{j=1}^{n} Enc(T[i,j])^{y_j^{(0)}} \quad (5)$$

for i = 1, . . . , n, and sends to customer. Then after receiving $Enc(T. y^{(0)})$ the customer decrypts and knows $T. y^{(0)}$ using his private key. He then updates the next approximation $y^{(1)} = T . y^{(0)} + c'$ via (4).similarly for the kth iteration, the customer sends the kth approximation $y^{(k+1)}$ to cloud. The cloud sends $Enc(T.y^{(k)})$ to the customer for the next update of $y^{(k+1)}$.The protocol continues until the result converges, as shown Algorithm 2 .

The following algorithm shows the protocol execution for the phase of Problem Solving

```
Algorithm 2: Iterative Problem Solving Phase
Data: transformed LP problem with input $\mathbf{c}'$ and Enc $(\mathbf{T})$
Result: solution $\mathbf{x}$ to the original LP problem $\mathbf{\Phi} = (\mathbf{A}, \mathbf{b})$
% $L$: maximum number of iterations to be performed;
% $\epsilon$: measurement of convergence point;
Begin
1 Customer picks $\mathbf{y}^{(0)} \in (Z_N)^n$ ;
  for ($k \leftarrow 0$ to $L$) do
2       Customer sends $\mathbf{y}^{(k)}$ to cloud ;
3       Cloud computes Enc ( $\mathbf{T}\ \mathbf{y}^{(k)}$ ) via Eq. (5) ;
4       Customer decrypts $\mathbf{T}\ \mathbf{y}^{(k)}$ via his private key ;
        if $||\mathbf{y}^{(k)} - \mathbf{y}^{(k+1)}|| \le \epsilon$ then
5           break with convergence point $\mathbf{y}^{(k+1)}$ ;
6   return $\mathbf{x} = \mathbf{y}^{(k+1)} - \mathbf{r}$ ;
```

To determine termination, the customer tests if

$$\left\| y^{(k)} - y^{(k+1)} \right\| \leq \epsilon \qquad (6)$$

C Result Verification

In many cases, a dishonest cloud server could damage the execution of protocol by either being acting lazy or intentionally damaging the computation result. So here we propose to design result verification methods to handle these two malicious behaviors. Here we denote $z^{(k)} = T \cdot y^{(k)}$ as the expected correct responses, and $z\hat{}^{(k)} = T \cdot \hat{y}^{(k)}$ as the actual received value from cloud server, where k = 1, 2, . . . ,*L*. After receiving the results customer does as shown in the fig 4.

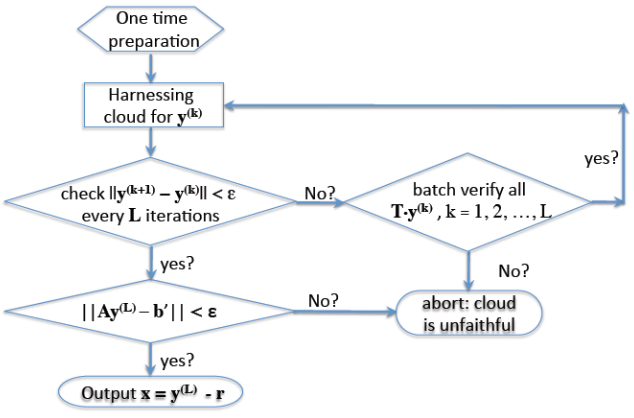
One time preparation
Harnessing cloud for $\mathbf{y}^{(k)}$
check $\|\mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}\| < \varepsilon$ every **L** iterations
No?
batch verify all $\mathbf{T}\cdot\mathbf{y}^{(k)}$, k = 1, 2, …, L
yes?
yes?
No?
$||\mathbf{A}\mathbf{y}^{(L)} - \mathbf{b}'|| < \varepsilon$
No?
abort: cloud is unfaithful
yes?
Output $\mathbf{x} = \mathbf{y}^{(L)} - \mathbf{r}$

Fig 4 Result verification process

1) Dealing with Indolent Opponent:

Computing the addition and multiplication over encrypted data could cost a lot of computational power, moreover the cloud server may not be willing to commit service level-agreed computing resources in order to save cost i.e. for the kth iteration, the opponent could simply reply the result $z^{(k-1)}$ of the previous (k -1)th iteration without computation. As a result, the customer who uses $z^{(k-1)}$ to update for the next $y^{(k+1)}$ will get the result $y^{(k+1)} = y^{(k)}$. This may result in incorrectly believing the solution of equation Ay =b′ is found. Thus, for the indolent opponent, only checking the (6) is not sufficient in order to convince the customer that the solution has converged. Another one step has to be executed as

$$\left\| A y^{(k+1)} - b' \right\| \leq \epsilon \qquad (7)$$

2) Dealing with Malevolent Adversary:

A malicious adversary can damage the whole protocol execution by returning incorrect answers, so here we design an efficient and effective method to detect such malicious behavior to ensure the result quality. By using the algebraic property of matrix-vector multiplication we design a method to test the correctness of all received answers $z\hat{}^{(k)} = T \cdot \hat{y}^{(k)}$, k = 1, 2, . . . *L* in only one batch. If suppose after L iterations, the solution still does not converge. The customer can initiate a Result Verification phase by randomly selecting *L* numbers, $\alpha_1, \alpha_2, \ldots, \alpha_L$,where each α_k is of *l*-bit length and $l < \log N$. Then he computes the linear combination θ over the $y^{(k)}$'s, which customer has provided in the previous k iterations, Next ,in order to test the correctness of all the intermediate results, $\{ z\hat{}^{(k)} = T \cdot \hat{y}^{(k)} \}$, which is received from cloud server, the customer checks if the following equation holds:

$$T \cdot \theta = \sum_{k=1}^{L} \alpha_k \cdot \hat{z}^{(k)} \qquad (8)$$

D Output Privacy Analysis

By using the proposed protocol the cloud server can see only the plain text of $y^{(k)}$, Enc(T) the encrypted matrix, and Enc(T. $y^{(k)}$) the encrypted vectors. Encrypted problem y of linear programming is the blinded version of original solution x and it is very secure to send the encrypted version of the plaintext, so that no information of x will be leaked to the server because r is kept private by the customer, hence for each individual linear equation a randomly picked vector is used to protect the privacy of the output.

E Input Privacy Analysis

Since the cloud server has no previous knowledge about the coefficient matrix no information leakage is possible especially when the problem size is very large. Also by using random scaling factor for each iteration in order to split the connection of two consecutive iterations of the protocol. Customer sends $y^{(k)}$ and also the scaling factor to the cloud instead of sending only $y^{(k)}$ in the kth iteration of problem solving phase, the customer only needs to decrypts the vector divides each component with scaling factor and updates the next approximation when cloud server sends the encrypted value for the next iteration one more random scaling factor is multiplied to this approximation and then sent to the cloud hence input privacy is protected.

VI. CONCLUSION

The problem of privacy assured delegation of massive LP computation workload in cloud computing different from previous work, here the computation outsourcing is based on iterative method. Security for the confidential data is provided via a semantically secure additive homomorphic encryption. The design requires a one-time repayable setup phase with $O(n^2)$ cost, and iterative algorithm execution only requires less local computing cost along with the benefits of easy to-implement and less memory requirement in practice. Our mechanism also provides efficient and effective dishonest detection scheme for result verification.

REFERENCES

[1] M. Armbrust et al., "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.
[2] C. Gentry, "Computing Arbitrary Functions of Encrypted Data," Comm. ACM, vol. 53, no. 3, pp. 97-105, 2010.
[3] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing," http://www.cloudsecurityalliance.org, 2009.
[4] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," EUROCRYPT: Proc. 17th Int'l Conf. Theory and Application of Cryptographic Techniques, pp. 223-238, 1999.

[5] B. Carpentieri, “Sparse Preconditioners for Dense Linear Systems from Electromagnetic Applications,” PhD dissertation, CERFACS, Toulouse, France, 2002.

[6] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, “Secure Ranked Keyword Search over Encrypted Cloud Data,” Proc. IEEE 30th Int’l Conf. Distributed Computing Systems (ICDCS), pp. 253-262, 2010.

[7] M. Atallah and K. Frikken, “Securely Outsourcing Linear Algebra Computations,” Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS), pp. 48-59, 2010.

[8] D. Benjamin and M.J. Atallah, “Private and Cheating-Free Outsourcing of Algebraic Computations,” Proc. Sixth Conf. Privacy ,Security, and Trust (PST), pp. 240-245, 2008.

[9] K. Nissim and E. Weinreb, “Communication Efficient Secure Linear Algebra,” Proc. Third Conf. Theory of Cryptography (TCC),pp. 522-541, 2006.

[10] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed. Soc. for Industrial and Applied Math., 2003.

[11] E. Kiltz, P. Mohassel, E. Weinreb, and M.K. Franklin, “Secure Linear Algebra Using Linearly Recurrent Sequences,” Proc. Fourth Conf. Theory of Cryptography (TCC), pp. 291-310, 2007.

[12] S. Goldwasser, Y.T. Kalai, and G.N. Rothblum, “Delegating Computation: Interactive Proofs for Muggles,” Proc. ACM Symp.Theory of Computing (STOC), pp. 113-122, 2008.