Volume 4, Issue 4, April 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Secure Outsourcing of Linear Computations into the Cloud

Sagarika Dev Roy, Srividhya Ganesan

Department of Computer Science and Engineering,
AMC Engineering College, Bangalore, India

Abstract— Computation outsourcing is one of the largest possibilities that the cloud enables. Clients with limited resources can utilize any necessary computing resources to outsource large-scale computations to the cloud. Despite the benefits of cloud computing, protection of the customers' confidential data is a major concern. This work deals with secure outsourcing of linear computations into the cloud and also solving it securely. Using the direct methods like Gaussian Elimination to large-scale linear computations is very expensive. Therefore, iterative method such as the Jacobi method is used to solve these linear equations, which is easier to implement. This method enables customers to find successive approximations to the linear equation solution iteratively and securely, where the sensitive input and output of the computation are kept private. For the method of detecting a malicious server, an efficient result verification mechanism is proposed.

Keywords—Confidential data, computation outsourcing, linear equations, cloud computing

I. INTRODUCTION

Outsourcing is a general procedure employed in the business world when the customer chooses to farm out a certain task to an agent. The reasons for the customer to outsource the task to the agent can be many, ranging from a lack of resources to perform the task locally to a deliberate choice made for financial or response time reasons. 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.

In spite of the tremendous benefits, there are many security concerns and challenges because the customers and cloud are not in the same trusted domain [1]. The data that are processed and generated during the computation in cloud are often confidential data. Moreover, since the operational details inside the cloud are not transparent enough to customers, so no guarantee is provided on the quality of the computed results from the cloud. For example, for computations which demand a large amount of resources, it is very likely that the cloud server (CS) can become "lazy" and might not evaluate the customer cannot tell the correctness of the answer. Therefore, the cloud is assumed to be not secure from the viewpoint of customers. Without providing a mechanism for secure computation outsourcing, i.e., to protect the sensitive input and output data and to validate the computation result integrity, it would be hard to expect customers to turn over control of their computing needs from local machines to cloud solely based on its economic savings.

This paper deals with secure outsourcing of large-scale systems of linear equations (LE), which are among the most popular algorithmic and computational tools in various engineering disciplines that analyze and optimize real-world systems. The storage requirements of the system coefficient matrix may easily exceed the available memory of the customer's computing device [2]. There are many real-world problems that would lead to very large-scale systems of linear equations with up to hundreds of thousands [3]. A typical double-precision 50,000×50,000 system matrix that can result from an electromagnetic application would easily occupy up to 20 GB storage space which challenges the computational power of low-end computing devices. Also, the execution time of a computer program depends not only on the number of operations it must execute, but also on the location of the data in the memory hierarchy [2].Therefore, solving large-scale problems on customer's devices can be impossible, due to huge Input/Output(IO) cost that is involved. Thus, resorting to cloud for such computation intensive tasks can be the only choice for customers with weak computing power, especially when the solution is demanded in a timely fashion.

Existing approaches do not address the asymmetry among the computational power possessed by cloud and the customer. Moreover, all these solutions are based on direct method for solving the LE, like the Gaussian elimination method [4] or the secure matrix inversion method in [5]. These approaches work well for small size problems, however it do not derive acceptable solution time for large-scale LE, due to the expensive cubic time computational burden for matrix-matrix operations and the huge IO cost on customer's weak devices. The proposed mechanism designs a secure mechanism of outsourcing LE via the Jacobi iterative method, where the solution is extracted via finding successive approximations to the solution until the required accuracy is obtained. Compared to direct method, an iterative method only demands relatively simpler matrix-vector operations with $O(n^2)$ computational cost, which is much easier to implement in practice and widely adopted for large-scale LE [3], [7]. This proposed mechanism also utilizes the additive

homomorphic encryption scheme, e.g., the Paillier cryptosystem [8]. Paillier cryptosystem is a probabilistic asymmetric algorithm for public key cryptography. It is used due to its flexibility for fitting larger plaintexts with a constant expansion ratio. For a linear system with $n \times n$ coefficient matrix, the proposed mechanism is based on a one-time amortizable setup with $O(n^2)$ cost. In each iterative algorithm execution, the proposed mechanism only incurs $O(n)$ local computational burden to the customer and asymptotically eliminates the expensive IO cost. To ensure computation result integrity, a very efficient cheating detection mechanism is also proposed to effectively verify the computation results by the cloud server from previous algorithm iterations with high probability.

II. LITERATURE REVIEW

A. Secure Computation Outsourcing

The theory based on Yao's garbled circuits [11] and Gentry's fully homomorphic encryption (FHE) scheme [12] would be impractical due to the extremely high complexity of FHE operation and the pessimistic circuit sizes that can hardly be handled. In [13], Atallah et al. give the first investigation of secure outsourcing of scientific computation. Their work explicitly allows private information leakage. Besides, the important case of result verification is not considered. In [9], Atallah et al. give a protocol design for secure matrix multiplication outsourcing. The design is built upon the assumption of two non-colluding servers and thus vulnerable to colluding attacks. Later on in [10], Atallah et al. give an improved protocol for secure outsourcing matrix multiplications based on secret sharing, which outperforms their previous work [9] in terms of single server assumption and computation efficiency. But due to secret sharing technique, all scalar operations in original matrix multiplication are expanded to polynomials, introducing significant communication overhead. Very recently, Wang et al. [14] give the first study of secure outsourcing of linear programming in cloud computing. Their solution is based on problem transformation, and has the advantage of bringing customer savings without introducing substantial overhead on cloud. However, those techniques involve cubic-time computational burden matrix-matrix operations, which the weak customer is not necessarily able to handle for large-scale problems.

B. Secure Two-party Computation

Securely solving LE has also been studied under the framework of SMC [11]. The work under SMC may not be well-suited for the computation outsourcing model, primarily because they generally do not address the computation asymmetry among different parties. Besides, in SMC no single involved party knows the entire problem input information, making result verification usually a difficult task. But in the proposed model, the fact that the customer knows all input information can be explicitly exploited and thus design efficient batch result verification mechanism. The most communication efficient work such as [4], [5] focus only on the direct method based solution, and thus are not necessarily able to tackle the large-scale problem. Troncoso-Pastoriza et al. [6] give the first study on iterative methods for collaboratively solving systems of linear equations under the SMC framework, which is the closest related work. However, their work can only support very limited number of iterative algorithm execution and cannot support reasonably large-scale systems of linear equations due to the continuing growth of the ciphertext in each of the iteration, which is not the case in the proposed design. In addition to the computation asymmetry issue mentioned previously, they only consider honest-but-curious model and thus do not guarantee that the final solution is correct under the possible presence of truly malicious adversary.

C. Computation Delegating and Cheating Detection

Detecting the unfaithful behaviors for computation outsourcing is not an easy task, even without consideration of input/output privacy. Verifiable computation delegation, where a computationally weak customer can verify the correctness of the delegated computation results from a powerful but untrusted server without investing too much resource, has found great interests in theoretical computer science community. Szajda et al. [15] propose methods to deal with cheating detection of other classes of computation outsourcing, including optimization tasks and Monte Carlo simulations. In [16], Du. et al. propose a method of cheating detection for general computation outsourcing in grid computing. The server is required to provide a commitment via a Merkle tree based on the results it computed. 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. However, all above schemes allow server actually see the data and result it is computing with, which is strictly prohibited in secure computation outsourcing model for data privacy. Thus, the problem of result verification essentially becomes more difficult, when both the input/output privacy is demanded.

III. PROBLEM STATEMENT

A. System Model

Suppose the cloud customer has a large-scale LE problem Ax=b which is required to be solved and is denoted by Φ=(A,b). Due to lack of computing resources, he cannot carry out the expensive ($O(n^{\rho})(2<\rho\leq 3)$) computation locally. Therefore the customer outsources the linear equation problem to the cloud server. For the protection of the data, the customer uses a secret key K to map Φ into an encrypted version Φ_K. Then, based on Φ_K, the customer starts the computation outsourcing protocol with CS, and harnesses the cloud resources in a privacy-preserving manner. The CS finds the answer of Φ_K, and after receiving the solution of encrypted problem Φ_K, the customer should be able to first verify the answer. If the answer is correct, he then uses the secret K to map the output into the desired answer for the original problem Φ. The computation outsourcing architecture involving cloud customer and CS is considered as illustrated in Fig.1.

Here, the customer needs to perform a one-time setup phase of encrypting the coefficient matrix with relatively costly $O(n^2)$ computation. But it is important to stress that this process can be performed under a trusted environment where the

weak customer with no sufficient computational power outsources it to a trusted party. The motivating example can be a military application where the customer has a one-time encryption process executed inside the military base by a trusted server, and then goes off into the field access only to untrusted CS.

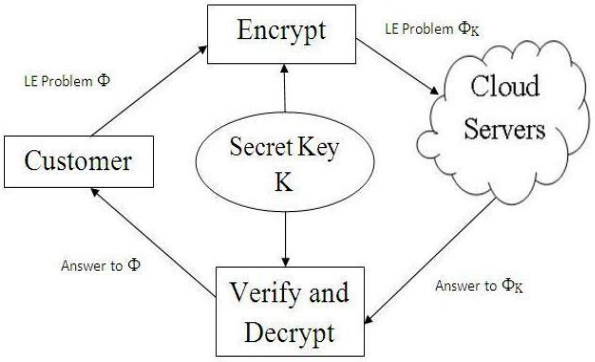

Fig. 1. Architecture of secure outsourcing of linear computations into the cloud.

So, it is assumed that CS is already in possession of the encrypted coefficient matrix, and the customer who knows the decryption key hopes to securely harness the cloud for on-demand computing outsourcing needs.

B. Methodologies

1) Jacobi Iterative Method:

In computational mathematics, an iterative method is a mathematical procedure that generates a sequence of improving approximate solutions for a class of problems. An iterative method is called convergent if the corresponding sequence converges for given initial approximations. In contrast, direct methods attempt to solve the problem by a finite sequence of operations. In the absence of rounding errors, direct methods would deliver an exact solution. Iterative methods are often the only choice for nonlinear equations. However, iterative methods are often useful even for linear problems involving a large number of variables where direct methods would be prohibitively expensive even with the best available computing power.

A system of linear equations is written as

$$Ax=b \tag{1}$$

where x is the $n\times1$ vector of unknowns, A is an $n\times n$(nonsingular) coefficient matrix, and b is an $n\times1$ right-hand side vector (so called constant terms).

Examples of iterative methods are Jacobi method, Gauss-Seidel method and the successive over-relaxation method. Without loss of generality, Jacobi iteration [7] is focused here for its simplicity. The method starts with the decomposition: A=D+R, where D is the diagonal component and R is the remaining matrix. The Equation (1) can be written as Ax=(D+R)x=b, and finally reorganized as: $x=-D^{-1}.R.x+D^{-1}.b$. According to the Jacobi method, an iterative technique can be used to solve the left hand side of this expression for $x^{(k+1)}$, using previous value for $x^{(k)}$ on the right hand side. If the iteration matrix is denoted by $T=-D^{-1}.R$ and $c=D^{-1}.b$, the above iterative equations can be represented as

$$x^{(k+1)}=T.x^{(k)}+c \tag{2}$$

2) Homomorphic Encryption:

Additive homomorphic property is used for a secure encryption scheme. Let x_1 and x_2 be two integers where $Enc(x_1+x_2)=Enc(x_1)*Enc(x_2)$ and $Enc(x_1*x_2)=Enc(x_1)^{x}{}_2$. In this implementation, the Paillier cryptosystem [8] is adopted. It is a probabilistic asymmetric algorithm for public key cryptography. For a vector $x=(x_1,x_2,\ldots,x_n)^T \in (\mathbb{Z}_N)^n$, Enc(x) can be used to denote the coordinate-wise encryption of x: $Enc(x)=(Enc(x_1), Enc(x_2),\ldots, Enc(x_n))^T$. For some $n\times n$ matrix T, where each of the component T[*i*,*j*] in T is from $\mathbb{Z}_N$, the component-wise encryption of T is denoted as Enc(T), and Enc(T)[*i*,*j*]=Enc(T[*i*,*j*]).

IV. THE FRAMEWORK AND BASIC MECHANISM

The framework for the proposed mechanism design is shown in this section and a basic mechanism is provided based on direct method. The basic mechanism fulfills the input/output privacy, but does not meet the efficiency requirement. The analysis of this basic solution gives insights and motivations on the main mechanism design based on iterative methods.

A. The General Framework

There are three phases: TransfProb, SolvProb, ResultVerify.

1) ***TransfProb****:* In this phase, cloud customer initializes a randomized key generation algorithm and transforms the LE problem into an encrypted form Φ_K via key *K* for phase SolvProb.
2) ***SolvProb****:* In this phase, cloud customer uses the encrypted form Φ_K of LE to start the computation outsourcing process. The protocol ends when the solution within the required accuracy is found.
3) ***ResultVerify****:* In this phase, the cloud customer verifies the encrypted result produced from cloud server, using the randomized secret key *K*. A correct output x to the problem is produced by decrypting the encrypted output. When the validation fails, then it means that the cloud server is cheating.

B. Basic Mechanism

The analysis of this basic solution gives motivations on the proposed mechanism design based on iterative methods. In the TransfProb phase, a random vector $r \in \mathbb{R}^n$ is taken as a secret keying material by the customer. Then, Equation (1) is rewritten as A(x+r)=b+Ar. Now, let y=x+r and b´=b+Ar, and Ay=b´. To hide the coefficient matrix A, a random invertible matrix Q is selected which has the same size as A. Left multiplying Q to both sides of Ay=b´ give us

$$A´y=b´´ \tag{3}$$

where A´=QA and b´´=Q(b+Ar). The customer can then start the SolvProb phase by outsourcing Φ_K=(A´,b´´) to the cloud, who solves Φ_K and sends back y. After verifying the correctness of y, the customer can derive the original x via x=y-r.

While achieving the input/output protection, this approach is not attractive for the following reasons:

1) The local problem transformation cost for matrix multiplication QA is $O(n^3)$, which is comparable to the cost of solving Ax=b[11]. Considering the extra cost of ResultVerify, there is no guaranteed computational saving for the customer.

2) The local cubic time cost can become prohibitively expensive when n goes large to the orders of hundreds of thousands. Besides, it violates the assumption that the customer cannot carry out expensive $O(n^\rho)(2<\rho\leq 3)$ computation locally.

V. THE PROPOSED MECHANISM

A. Transformation of the Problem

The customer who has the coefficient vector b and seeks for solution x satisfying Ax=b cannot directly start the SolvProb Phase with cloud for the protection of the result x. Therefore, a transformation mechanism is needed which allows the customer to properly hide such information first. Similar to the basic mechanism, in the TransfProb phase, the customer will pick a random vector $r \in \mathbb{R}^n$ as his secret keying material, and rewrites Equation(1) as the new LE problem

$$Ay=b´ \tag{4}$$

where y=x+r and b´=b+Ar. The solution x to Equation(1) can be found by solving a transformed LE problem in Equation(4), and vice versa. At this point, both the output x and input tuple b will be hidden by the random vector r. Next, Equation(4) is rewritten in an iterative form similar as Equation(2):

$$y^{(k+1)}=T.y^{(k)}+c´ \tag{5}$$

where $T=-D^{-1}R$, $c´=D^{-1}.b´$, and A=D+R. Now the problem input Φ=(A,b) is changed to tuple Φ_K=(T,c´), where T has already been encrypted and stored as Enc(T) at cloud, and c´ is just a randomly masked version of b via random n×1 vector r. The output x is also masked by y=x+r. This problem transformation only requires locally two matrix-vector multiplications: b´=b+Ar and $c´=D^{-1}.b´$ with n^2+n scalar multiplications. When n goes large, expensive IO cost at customer device might downgrade the performance of such operations. By comparing Equation(2) and Equation(5), it is easy to see that this transformation does not affect the matrix of A (or T), which gives the advantage of reusing.

B. Solution of the Problem

The goal here is to harness the cloud for the most expensive computation securely, i.e., the matrix-vector multiplication $T.y^{(k)}$ in Equation(5) for each algorithm iteration, k=1,2,….,L.

1. In the first iteration, the customer first starts the 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.

2. The cloud server which possess of the encrypted matrix Enc(T) already, computes the value $Enc(T.y^{(0)})$ by using the homomorphic property of the encryption:

$$Enc(T.\ y^{(0)})[i] = Enc(\textstyle\sum_{j=1}^{n} T[i,j],y_j^{(0)}) = \prod_{j=1}^{n} Enc(T[i,j])^{yj(0)} \tag{6}$$

for i=1,…,n, and sends $Enc(T.y^{(0)})$ to customer.

3. After receiving $Enc(T.y^{(0)})$, the customer decrypts and gets $T.y^{(0)}$ using his private key. He then updates the next approximation $y^{(1)}=T.y^{(0)}+c´$ via Equation(5).

For the kth iteration, it follows that the customer sends the kth approximation $y^{(k)}$ 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. The customers computation overhead is to decrypt the vector of $Enc(T.y^{(k)})$ and it takes $O(n)$ time complexity and does not require expensive IO cost.

C. Convergence Analysis

It is needed to determine whether and when the iteration will converge. Since it is assumed that the matrix A ensures convergence, the customer tests if

$$||y^{(k)}-y^{(k+1)}|| \leq \epsilon \tag{7}$$

for some small enough ϵ>0. And the termination point y(k+1) will help to get the final result x via=$y^{(k+1)}$-r. the local computation is $O(n)$.

D. Input/Output Privacy Analysis

1) Output Privacy Analysis :

The cloud server must be able to see only the plaintext of $y^{(k)}$, the encrypted version of matrix Enc(T), and encrypted vectors Enc(T.$y^{(k)}$), k=1,2,…,L. Since y is a blinded version of original solution x, it is safe to send y to the cloud in plaintext. No information of x would be leaked as long as r is kept secret by the customer.

2) Input Privacy Analysis:

Some knowledge about the input tuple Φ_k=(T,c´) could be implicitly leaked through the protocol execution itself. The reason is as follows: for each two consecutive iterations of the protocol, namely, the k^{th} and the $(k+1)^{th}$, the cloud server sees actually the plaintext of both $y^{(k)}$ and $y^{(k+1)}$. Thus, a "clever" cloud server could initiate a system of linear equations via Equation (5) and attempts to learn the unknown components of T and c´. More specifically, for the total L iterations, the cloud server could establish a series of $(L-1)\times n$ equations from $y^{(k)}$, k=0,1,…,L-1, to solve n^2+n unknowns of T and c´.

E. Detection of Cheating Clouds

In many cases, an unfaithful cloud server could deliberately destroy the protocol execution by being lazy and corrupts the computation result. It is proposed to design result verification methods to handle this kind of a behavior. The goal is to verify the correctness of the solution by using as few as possible expensive matrix-vector multiplication operations.

Since computing the addition and multiplication over encrypted domain could cost a lot of computational power, the cloud server might not be willing to commit service level-agreed computing resources in order to save cost for example, for the k^{th} iteration, the adversary could simply give the result of the previous $(k-1)^{th}$ iteration without computation.

As a result, the customer who uses this result to update for the next $y^{(k+1)}$ will get the new result as $y^{(k+1)}=y^{(k)}$. Consequently, he may be incorrectly led to believe the solution of equation Ay=b´ is found. Thus, for the malicious adversary, only checking the Equation(7) is not sufficient to convince the customer that the solution has converged. According to Equation(4) one further step has to be executed as

$$||Ay^{(k+1)}-b'|| \leq \epsilon \tag{8}$$

This step does not have to be executed within all the iterations. It only needs to be tested after the test on $y^{(k)}$ and $y^{(k+1)}$ via Equation(7) is passed. If Equation(7) is not passed, it means $y^{(k+1)}$ is not the convergence point yet. On the other hand, if Equation(7) is successfully passed, it can then initiate the test of Equation(8). If Equation(8) holds, it can be said that the final solution is found, which is x=$y^{(k+1)}$-r. If it doesn't, it can be said that the cloud server is cheating (being lazy). In either case, this matrix-vector multiplication of Equation(8) only needs to be executed at most once throughout the protocol execution. The method for cheating detection indeed achieves as few as possible expensive matrix-vector multiplication operation. As a result, the overall design must be able to eliminate the expensive IO cost asymptotically on customer.

VI. CONCLUSION

This paper formulates the problem of securely outsourcing linear equations via the Jacobi iterative method and provides mechanism designs fulfilling input/output privacy, cheating resilience and efficiency. The proposed mechanism brings computational savings. Within each iteration, it incurs $O(n)$ computation burden for the customer and demands no unrealistic IO cost, while solving the linear equations locally incurs $O(n^2)$ per iteration cost in terms of both time and memory requirements. It also allows the customers to verify all results of previous iterations from cloud in one batch. It ensures both the efficiency advantage and robustness of the design.

REFERENCES

[1] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing," http://www.cloudsecurityalliance. org, 2009.

[2] K. Forsman, W. Gropp, L. Kettunen, D. Levine, and J. Salonen, "Solution of Dense Systems of Linear Equations Arising from Integral-Equation Formulations," IEEE Antennas and Propagation Magazine, vol. 37, no. 6, pp. 96-100, Dec. 1995.

[3] A. Edelman, "Large Dense Numerical Linear Algebra in 1993: The Parallel Computing Influence," Int'l J. High Performance Computing Applications, vol. 7, no. 2, pp. 113-128, 1993.

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

[5] 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.

[6] J.R. Troncoso-Pastoriza, P. Comesana, and F. Perez-Gonzalez, "Secure Direct and Iterative Protocols for Solving Systems of Linear Equations," Proc. First Int'l Workshop Signal Processing in the EncryptEd Domain (SPEED), pp. 122-141, 2009.

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

[8] 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.

[9] 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.

[10] M. Atallah and K. Frikken, "Securely Outsourcing Linear Algebra Computations," Proc. Fifth ACM Symp. Information, Computer and Comm. Security (ASIACCS), pp. 48-59, 2010.
[11] A.C.-C. Yao, "Protocols for Secure Computations (Extended Abstract)," Proc. IEEE 23rd Symp. Foundations of Computer Science (FOCS), pp. 160-164, 1982.
[12] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proc. 41st Ann. ACM Symp. Theory of computing (STOC), pp. 169-178, 2009.
[13] 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.
[14] C. Wang, K. Ren, and J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," Proc. IEEE INFOCOM, pp. 820-828, 2011.
[15] D. Szajda, B.G. Lawson, and J. Owen, "Hardening Functions for Large Scale Distributed Computations," Proc. IEEE Symp. Security and Privacy, pp. 216-224, 2003.
[16] W. Du, J. Jia, M. Mangal, and M. Murugesan, "Uncheatable Grid Computing," Proc. 24th Int'l Conf. Distributed Computing Systems (ICDCS), pp. 4-11, 2004.

