New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations

Xiaofeng Chen, Xinyi Huang, Jin Li, Jianfeng Ma, Wenjing Lou, and Duncan S. Wong

***Abstract*— With the rapid development in availability of cloud services, the techniques for securely outsourcing the prohibitively expensive computations to untrusted servers are getting more and more attentions in the scientific community. In this paper, we investigate secure outsourcing for large-scale systems of linear equations, which are the most popular problems in various engineering disciplines. For the first time, we utilize the sparse matrix to propose a new secure outsourcing algorithm of large-scale linear equations in the fully malicious model. Compared with the state-of-the-art algorithm, the proposed algorithm only requires (optimal) one round communication (while the algorithm requires *L* rounds of interactions between the client and cloud server, where *L* denotes the number of iteration in iterative methods). Furthermore, the client in our algorithm can detect the misbehavior of cloud server with the (optimal) probability 1. Therefore, our proposed algorithm is superior in both efficiency and checkability. We also provide the experimental evaluation that demonstrates the efficiency and effectiveness of our algorithm.**

***Index Terms*— Cloud computing, outsource-secure algorithms, system of linear equations.**

I. Introduction

CLOUD Computing, the long dreamed vision of computing as a utility, enables convenient and on-demand network access to a centralized pool of configurable computing resources that can be rapidly deployed with great efficiency and minimal management overhead [46], [47]. Cloud computing has plenty of benefits for real-world applications such as on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing, outsourcing, etc. In the outsourcing computation paradigm, the users with resource-constraint devices can outsource heavy computation workloads into the cloud server and enjoy the unlimited computing resources in a pay-per-use manner. As a result, the enterprises and individuals can avoid large capital outlays in hardware/software deployment and maintenance.

Despite the tremendous benefits, outsourcing computation also inevitably suffers from some new security challenges [42], [50]. Firstly, the computation tasks often contain some sensitive information that should not be exposed to the (semi-trusted) cloud servers. Therefore, the first security challenge is the *privacy* of the outsourcing computation: the curious cloud servers should not learn anything about what it is actually computing (including the secret inputs and outputs). We argue that the traditional encryption technique can only provide a partial solution to this problem since it is very difficult to perform meaningful computations over the ciphertext. Though the fully homomorphic encryption could be a potential solution, the existing schemes are not practical yet. Secondly, the semi-trusted cloud servers may return an invalid result. For example, the servers might contain a software bug that will fail on a constant number of invocation. Moreover, the servers might decrease the amount of the computation due to financial incentives and then return a computationally indistinguishable (invalid) result. Therefore, the second security challenge is the *checkability* of the outsourcing computation: the outsourcer should have the ability to detect any failures if the cloud servers misbehave. Trivially, the test procedure should never be involved in some other complicated computations since the computationally limited devices may be incapable to accomplish a complicated test. At the very least, it must be far more *efficient* than accomplishing the computation task itself (recall the motivation for outsourcing computations).

The large-scale system of linear equations $\mathbf{Ax} = \mathbf{b}$ [8], [23] is one of the most basic algebraic problems in the scientific community. In practice, there are many real world problems that would lead to very large-scale and extremely dense systems of linear equations with up to hundreds of thousands or even millions unknown variables. For example, a typical double-precision $50,000 \times 50,000$ system matrix resulted from electromagnetic application would easily occupy up to 20 GBytes storage space. Hence, the storage requirements of the system coefficient matrix may easily exceed the available

Manuscript received December 19, 2013; revised May 6, 2014 and October 10, 2014; accepted October 10, 2014. Date of publication October 16, 2014; date of current version December 5, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61272455, Grant 61472083 and Grant 61472091, in part by the Doctoral Fund, Ministry of Education, China, under Grant 20130203110004 and Grant 20123503120001, in part by the Distinguished Young Scholars Fund, Department of Education, Fujian Province, under Grant JA13062, in part by the Fok Ying Tung Education Foundation under Grant 141065, in part by the Program for New Century Excellent Talents in University under Grant NCET-13-0946, in part by the Program for New Century Excellent Talents in Universities of Fujian under Grant JA14067, in part by the Fundamental Research Funds for the Central Universities under Grant BDY151402, and in part by the 111 Project, China, under Grant B08038. The work of W. Lou was supported by the U.S. National Science Foundation under Grant CNS-1217889. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sen-Ching S. Cheung. *(Corresponding author: Xiaofeng Chen.)*

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710065, China (e-mail: xfchen@xidian.edu.cn).

X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China (e-mail: xyhuang81@gmail.com).

J. Li is with the School of Computer Science, Guangzhou University, Guangzhou 510006, China (e-mail: lijin@gzhu.edu.cn).

J. Ma is with the School of Computer Science and Technology, Xidian University, Xi'an 710065, China (e-mail: jfma@mail.xidian.edu.cn).

W. Lou is with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061 USA (e-mail: wjlou@vt.edu).

D. S. Wong is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: duncan@cityu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2014.2363765

memory of the customer's computing device such as a modern portable laptop. Plenty of researchers have devoted considerable amount of effort on seeking efficient algorithms for the task. Recently, Wang et al. [48] proposed a secure outsourcing mechanism for large-scale linear equations based on the iterative methods. However, one fatal disadvantage of Wang et al.'s scheme is multi-round communication between the client C and server S. More precisely, it may require dozens of (or even hundreds of) iterations for different matrix $\mathbf{A}$ by using the iteration methods (we present some examples in Table 4). As a result, it also requires dozens (or even hundreds) round of communications between C and S. Therefore, the scheme could be impractical for real-world applications.

A. Related Works

There are also plenty of research work on the securely outsourcing computations in the past decades. Abadi et al. [2] first proved the impossibility of secure outsourcing an exponential computation while locally doing only polynomial time work. Therefore, it is meaningful only to consider outsourcing expensive polynomial time computations.

The theoretical computer science community has devoted considerable attention to the problem of how to securely outsource different kinds of expensive computations. Atallah et al. [3] presented a framework for secure outsourcing of scientific computations such as matrix multiplications and quadrature. However, the solution used the disguise technique and thus allowed leakage of private information. Atallah and Li [4] investigated the problem of computing the edit distance between two sequences and presented an efficient protocol to securely outsource sequence comparisons to two servers. Recently, Blanton et al. proposed a more efficient scheme for secure outsourcing sequence comparisons [11]. Benjamin and Atallah [6] addressed the problem of secure outsourcing for widely applicable linear algebra computations. However, the proposed protocols required the expensive operations of homomorphic encryptions. Atallah and Frikken [1] further studied this problem and gave improved protocols based on Shamir's secret sharing. Some other works [36], [39] also used Shamir's secret sharing to perform homomorphic computations over cloud. Trivially, the protocols based on secret sharing require at least two non-colluding servers. Wang et al. [49] presented efficient mechanisms for secure outsourcing of linear programming computations. However, the solution requires matrix-matrix operations (cubic-time computational burden). Recently, Wang et al. [48] proposed a secure outsourcing mechanism for solving large-scale systems of linear equations based on the iterative methods. However, it requires multi-round interactions between the client and the cloud server and thus is impractical.

In the cryptographic community, Chaum and Pedersen [15] firstly introduced the notion of wallets with observers, a piece of secure hardware installed on the client's computer to perform some expensive computations. Hohenberger and Lysyanskaya [33] proposed the first outsource-secure algorithm for modular exponentiations based on the two previous approaches of precomputation [14], [40], [43] and server-aided computation [7], [27], [38], [51]. Recently, Chen et al. [21] proposed more efficient outsource-secure algorithms for (simultaneously) modular exponentiation in the two untrusted program model. Chevallier-Mames et al. [22] presented the first algorithm for secure delegation of elliptic-curve pairings based on an untrusted server model. Besides, the outsourcer could detect any failures with probability 1 if the server misbehaves. However, an obvious disadvantage of the algorithm is that the outsourcer should carry out some other expensive operations such as scalar multiplications and exponentiations.

Since the servers (or workers) are not trusted by the outsourcers, Golle and Mironov [29] first introduced the concept of ringers to solve the trust problem of verifying computation completion. The following researchers focused on the other trust problem of retrieving payments [5], [17]–[20], [45]. Besides, Gennaro et al. [25] first formalized the notion of verifiable computation and presented a verifiable computation scheme for any function. However, it is inefficient for practical applications due to the complicated fully homomorphic encryption techniques. Therefore, plenty of researchers investigated verifiable computation for specific functions in order to obtain much more efficient protocols [9], [12], [13], [26], [28], [34], [35], [37].[1] Benabbas et al. [10] presented the first practical verifiable computation scheme for high degree polynomial functions. Green et al. [24] proposed new methods for efficiently and securely outsourcing decryption of attribute-based encryption (ABE) ciphertexts. Based on this work, Parno et al. [41] showed a construction of a multi-function verifiable computation scheme.

B. Our Contribution

In this paper, we propose a new secure outsourcing algorithm for large-scale systems of linear equations $\mathbf{Ax} = \mathbf{b}$. Our proposed algorithm works with a single cloud server and the server is assumed to be lazy, curious, and dishonest (fully malicious model). Compared with the state-of-the-art algorithm [48], the proposed algorithm is superior in both efficiency and checkability. Our contributions are three folds:

1) For the first time, we utilize the sparse matrix to investigate securely outsourcing for large-scale systems of linear equations. Our algorithm is suitable for *any* nonsingular dense matrix $\mathbf{A}$. However, in algorithm [48], $\mathbf{A}$ must be a strictly diagonally dominant matrix for convergence.
2) Our proposed algorithm only requires (**optimal**) 1 round communication between the client C and server S, while algorithm [48] requires L interactions due to the iterative method.
3) In the proposed algorithm, the client C can detect the misbehavior of server S with the (**optimal**) probability 1. Surprisingly, we use neither the complicated knowledge proof techniques nor the boolean garbled circuits.

[1] We argue that it is inappropriate to claim these protocols which do not use homomorphic encryption are absolutely superior to Gennaro et al.'s one [25] since they have limited application scope for specific functions.

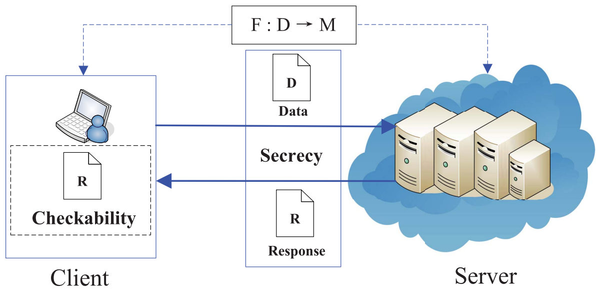


Fig. 1. Outsourcing Computation Architecture.

The computational complexity for the verification is still $O(n^2)$.

C. Organization

The rest of the paper is organized as follows: Some security definitions for outsourcing computation are given in Section II. The proposed new outsource-secure algorithm of linear equations and its security analysis are given in Section III. The experimental evaluation of the proposed algorithm is given in Section IV. Finally, conclusions will be made in Section V.

II. Security Model and Definitions

A. Security Model

We consider an outsourcing computation architecture in which an honest while resources-constrained client C wants to outsource an expensive computation task $\mathsf{F} : D \to M$ to a cloud server S, as illustrated in Fig. 1. Trivially, S is not fully trusted by C. Without loss of generality, we say that (C, S) correctly implements F if $\mathsf{F}(x) = C^S(x)$ for $\forall\ x \in D$, where C^S means that C is given oracle access to S that records all of its computation over time.

Golle and Mironov [29] first introduced the "lazy-but-honest" model for the *inversion of one-way function* class of outsourcing computations. As a rational economic agent, S will try to minimize the amount of work it needs to perform in order to retrieve the payment, while it will honestly provide the computation results to C. It is reasonable to assume that S is honest in this kind of outsourcing computation model. Actually, given an invalid computation result, C can detect it immediately with probability 1 since the verification of the result is equivalent to compute the one-way functions.

Another well-known model is the "honest-but-curious" model (originally called the semi-honest model), firstly introduced by Goldreich et al. [30]. In this model, both the parties are guaranteed to properly execute a prescribed protocol, but, at the end of it, one party can use its own view of the execution to infer about the other's input. Therefore, S will also honestly send the computation results to C. However, S will try his best to retrieve some sensitive information such as the secret input/output of C. Note that the computation results (i.e., the output of S) are different from the output of C (i.e., the real computation aim of C).

We can view S in the above two models as a passive adversary. In many applications, we must consider the case that S is an active attacker. For example, S can intentionally send a computationally indistinguishable (invalid) result to C. That is, S may be not only lazy and curious, but also dishonest. This is the strongest adversarial model, and we name it the "fully malicious model." It seems to be very difficult to design secure and efficient outsourcing algorithms for generic computations in the full malicious model.

Hohenberger and Lysyanskaya [33] first introduced a weaker model called "two untrusted program model" for outsourcing exponentiations modulo a large prime. In the two untrusted program model, there are two non-colluding servers S_1 and S_2 and we assume at most one of them is adversarial while we cannot know which one. Besides, the misbehavior of the dishonest server can be detected with an overwhelming probability. Recently, Canetti et al. [16] introduced the so-called "refereed delegation of computation model," where the outsourcer delegates the computation to n servers under the assumption that at least one of the servers is honest. Trivially, two untrusted program model can be viewed as a special case of refereed delegation of computation model for $n = 2$.

B. Formal Definition

Gennaro et al. [25] presented a formal definition for securely outsourcing computation.

Definition 1: A securely outsourcing computation scheme consists of a five-tuple (**KeyGen**, **ProbGen**, **Compute**, **Verify**, **Solve**):

1) **KeyGen**$(\mathsf{F}, k) \to (PK, SK)$: Given a security parameter k, the randomized key generation algorithm generates a public key PK that encodes the target function F, and a corresponding secret key SK which is kept private by the client C.
2) **ProbGen**$_{SK}(x) \to (\sigma_x, \tau_x)$: The problem generation algorithm uses the secret key SK to encode the function input x as a public value σ_x which is given to the server S to compute with, and a secret value τ_x which is kept private by the client C.
3) **Compute**$_{PK}(\sigma_x) \to \sigma_y$: Given the public key PK and the encoded input σ_x, the server computes an encoded version σ_y of the output $y = \mathsf{F}(x)$.
4) **Verify**$_{SK}(\sigma_y) \to 1 \cup 0$: On input the secret key SK, and the encoded value σ_y, the verification algorithm outputs 1 if σ_y is valid; Otherwise, outputs 0.
5) **Solve**$_{SK}(\tau_x, \sigma_y) \to y$: On input the secret key SK, the secret "decoding" τ_x, and the encoded value σ_y, the solving algorithm outputs the computation result $y = \mathsf{F}(x)$.

C. Security Requirements

In the following, we introduce some security requirements for outsourcing computation.

The first requirement is the *privacy* for the input/output of the computation task. Informally, it means that the server cannot learn anything from its interaction in the protocol in the sense of indistinguishability argument.

Definition 2 (Privacy [25]): Given a security parameter k, a pair of algorithms (C, S) is said to be privacy for the input/output of F if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$,

$$\text{Adv}_A^{C^S}(\mathsf{F}, k) \leq \text{negl}(k),$$

where $\text{Adv}_A^{C^S}(\mathsf{F}, k) = |\Pr[b = b'] - \frac{1}{2}|$ is defined as the advantage of A in the experiment as follows:

$$\begin{aligned}
(PK, SK) &\overset{R}{\leftarrow} \mathbf{KeyGen}(\mathsf{F}, k);\\
(x_0, x_1) &\leftarrow A^{\mathbf{PubProbGen}_{SK}(\cdot)}(PK)\\
(\sigma_0, \tau_0) &\leftarrow \mathbf{ProGen}_{SK}(x_0);\\
(\sigma_1, \tau_1) &\leftarrow \mathbf{ProGen}_{SK}(x_1);\\
b &\overset{R}{\leftarrow} \{0, 1\};\\
b' &\leftarrow A^{\mathbf{PubProbGen}_{SK}(\cdot)}(PK, x_0, x_1, \sigma_b)
\end{aligned}$$

During the above experiment, the adversary $\mathcal{A}$ is allowed to request the encoding of any input he desires. The oracle $\mathbf{PubProbGen}_{SK}(x)$ calls $\mathbf{ProGen}_{SK}(x)$ to obtain (σ_x, τ_x) and returns only the public part σ_x. Trivially, the output of $\mathbf{PubProbGen}_{SK}(x)$ is probabilistic.

The second requirement is the *efficiency* of outsourcing algorithms. That is, the local computation done by the client C should be substantially less than that to accomplish the original computation by itself (i.e., without outsourcing).

Definition 3 (α-Efficiency [33]): A pair of algorithms (C, S) is said to be an α-efficient implementation of F if (1) C^S correctly implements F and (2) for $\forall\ x \in D$, the running time of C is no more than an α-multiplicative factor of the running time of F.

Finally, the output of outsourcing algorithms can be *checked* for correctness. More precisely, the invalid output given by any malicious server can not pass the verification and the client C will detect the error with a non-negligible probability.

Definition 4 (β-Checkability [33]): A pair of algorithms (C, S) is said to be a β-checkable implementation of F if (1) C^S correctly implements F and (2) for $\forall\ x \in D$, if a malicious S' deviates from its advertised functionality during the execution of $C^{S'}(x)$, C will detect the error with probability no less than β.

III. New Outsourcing Algorithm of Linear Equations

A. High Description

The problem for securely outsourcing large-scale systems of linear equations can be formulated as follows: The client C seeks for the solution to a large-scale system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a real coefficient matrix with rank n, and $\mathbf{b} \in \mathbb{R}^n$ is a coefficient vector. Due to the lack of computing resources, C could be infeasible to carry out such expensive computation as $O(n^\rho)$ for $2 < \rho \leq 3$ locally. Therefore, C will outsource the computation workloads to cloud server S in a pay-per-use manner. In this paper, we only focus on general nonsingular dense matrices $\mathbf{A}$, while our proposed solution is also applicable to sparse matrices. While for the case of (extremely) sparse matrices, C may be able to compute the solution of linear equations efficiently with other methods.

In this section, we propose a new algorithm **LE** for securely outsourcing large-scale systems of linear equations in the fully malicious model. That is, the computation is delegated to only one server who may be lazy, curious, and dishonest. Our main trick is that we use two random *sparse* matrices to hide $\mathbf{A}$. More precisely, C chooses two random sparse matrices $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{n\times n}$ and computes $\mathbf{T} = \mathbf{MAN}$. Due to the random blinding technique, we can prove that the proposed outsourcing algorithm satisfies the privacy for $\mathbf{A}$, $\mathbf{b}$ and the solution $\mathbf{x}$. Besides, it does not require any special encryption techniques with the homomorphic property. We argue that the complexity to compute $\mathbf{T}$ is $O(n^2)$ since both $\mathbf{M}$ and $\mathbf{N}$ are sparse matrices.

Our proposed algorithm only requires one round communication between C and S. Furthermore, C can detect the misbehavior of S with the probability 1. The computational complexity for the verification is still $O(n^2)$. This is due to the observation that it requires at most n^2 multiplications to compute $\mathbf{Ty}$ for any (even totally dense) matrix $\mathbf{T}$ and any coefficient vector $\mathbf{y}$.

Remark 1: It is well-known that the computational complexity for two dense matrices multiplication is $O(n^3)$. In the following, we consider the case that a sparse matrix $\mathbf{M}$ multiplies a dense one $\mathbf{A}$, *i.e.*, $\mathbf{MA}$. Without loss of generality, we assume that there are at most λ non-zero elements for each row of $\mathbf{M}$, where $\lambda << n$. In the real applications, we can let $\lambda = 10$ for the case that $\mathbf{M}$ and $\mathbf{A}$ are both $50,000 \times 50,000$ matrices. Obviously, it takes at most λn^2 multiplications to calculate $\mathbf{MA}$. That is, the computational complexity is $O(n^2)$ since the constant $\lambda << n$. Thus, C can efficiently compute $\mathbf{MAN}$ and the complexity is still $O(n^2)$.

B. Outsourcing Algorithm

The input of **LE** is a coefficient vector $\mathbf{b} \in \mathbb{R}^n$ and a coefficient matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. The output of **LE** is a coefficient vector $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{Ax} = \mathbf{b}$. The proposed algorithm **LE** is given as follows:

1) **KeyGen**: To implement this functionality, C firstly picks a random blinding coefficient vector $\mathbf{r} \in \mathbb{R}^n$ and two random sparse matrices $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{n\times n}$. Note that $(\mathbf{M}, \mathbf{N}, \mathbf{r})$ must be kept secret by C.
2) **ProbGen**: C firstly computes $\mathbf{c} = \mathbf{Ar} + \mathbf{b}$. Trivially, the original linear equations can be rewritten as $\mathbf{A}(\mathbf{x} + \mathbf{r}) = \mathbf{c}$. Then, C computes $\mathbf{T} = \mathbf{MAN}$ and $\mathbf{d} = \mathbf{Mc}$. Without loss of generality, we denote $\mathbf{y} = \mathbf{N}^{-1}(\mathbf{x}+\mathbf{r})$, where $\mathbf{N}^{-1}$ is the inverse of matrix $\mathbf{N}$. Note that in our algorithm, *no* party needs to compute $\mathbf{N}^{-1}$. It appears here only for representing the form of $\mathbf{y}$. In fact, if $\mathbf{N}^{-1}$ had to be computed, the algorithm would no longer be efficient as the time and computational complexities incurred by computing $\mathbf{N}^{-1}$ would be very undesirable. Please refer to **Remark 2** below for more discussions.
   Note that
   $$\mathbf{Ty} = \mathbf{MAN} \cdot \mathbf{N}^{-1}(\mathbf{x} + \mathbf{r}) = \mathbf{MA}(\mathbf{x} + \mathbf{r}) = \mathbf{Mc} = \mathbf{d}.$$
3) **Compute**: C sends $\mathbf{T}$ and $\mathbf{d}$ to S, and S responds with the solution $\mathbf{y}$ such that $\mathbf{Ty} = \mathbf{d}$.

4) **Verify**: C verifies whether the equations $\mathbf{Ty} = \mathbf{d}$ hold. If not, C outputs 0 and claims the misbehavior of S.
5) **Solve**: If **Verify** $= 1$, C computes $\mathbf{x} = \mathbf{Ny} - \mathbf{r}$.

Remark 2: Note that no party needs to compute $\mathbf{N}^{-1}$ and this is very important in our algorithm. The reason are two folds: Firstly, though $\mathbf{N}$ is a sparse matrix, the inverse matrix $\mathbf{N}^{-1}$ may be extremely dense. As a result, the computation and storage cost for $\mathbf{N}^{-1}$ will be very expensive. Secondly, the computation complexity of $\mathbf{N}^{-1}$ is $O(n^3)$ if we use the naive Gaussian elimination or Gauss-Jordan elimination method.[2] In some scenario, it is even comparable to solving linear equations. Trivially, this contradicts with the original aim of outsourcing computation (recall that C cannot carry out such expensive computation as $O(n^\rho)$ for $2 < \rho \leq 3$).

Remark 3: Note that the sparse matrices $\mathbf{M}$ and $\mathbf{N}$ must be invertible (also called nonsingular). By Lévy-Desplanques Theorem (see [32, Th. 6.1.10]), we know that a strictly diagonally dominant matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is definitely nonsingular. Therefore, in the real applications, we could choose sparse and row diagonally dominant matrix $\mathbf{A}$ such that $\sum_{j\neq i} |a_{ij}| < |a_{ii}|$ for all $1 \leq i \leq n$.

Remark 4: Given $\mathbf{T}$ and $\mathbf{d}$, S can solve the systems of linear equations $\mathbf{Ty} = \mathbf{d}$ in any desired methods such as elimination methods, decomposition (factorization) methods, iterative methods, etc. However, as pointed out in [8], for the systems comprising hundreds of millions or even billions of equations in as many unknowns, iterative methods may be the only option available (unless the iteration does not converge). Furthermore, we argue that it does not require the interactive protocol between C and S in our proposed solution. As a result, S can efficiently compute $\mathbf{y}$ using the iterative methods.

Remark 5: Note that $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{r}$ in our scheme can be used only one time and thus have to be generated each time for different linear equations. However, as shown in Table 3, **KeyGen** in our algorithm is extremely fast.

Moreover, in order to prevent the brute force attack (i.e., the adversary $\mathcal{A}$ systematically enumerates all possible candidate solutions of the problem and checks one by one until the correct one is found), the size of non-zero elements of $\mathbf{M}$ and $\mathbf{N}$ should be at least 80 bits for medium security. Similarly, each component of the vector $\mathbf{r}$ is also 80 bits. For long-term security, they should be 128 bits or more. Currently, if a supercomputer could check a billion billion (10^{18}) keys per second, then breaking a 128-bit key space by brute force requires about 3×10^{12} years!

C. Security Analysis

Theorem 1: In the fully malicious model, the algorithms (C, S) are privacy for $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{x}$.

Proof: We first prove the privacy for input $\mathbf{b}$ and output $\mathbf{x}$ of **LE**. Note that the adversary $\mathcal{A}$ can only know $\mathbf{T}$ and $\mathbf{d}$ throughout the whole algorithm **LE**. Besides, we have $\mathbf{b} = \mathbf{M}^{-1}\mathbf{d} - \mathbf{Ar}$, and $\mathbf{x} = \mathbf{Ny} - \mathbf{r}$. Since $\mathbf{r}$ is a random blinding coefficient vector in $\mathbb{R}^n$, both $\mathbf{b}$ and $\mathbf{x}$ are blinded by $\mathbf{r}$ in the sense of indistinguishability.

We then prove the privacy for input $\mathbf{A}$ of **LE**. Let $\mathbf{M} = (m_{ij})$, $\mathbf{N} = (n_{ij})$, $\mathbf{M}' = (m'_{ij})$, and $\mathbf{N}' = (n'_{ij})$ be four random nonsingular sparse matrices generated by C. Given two nonsingular dense matrices $\mathbf{A} = (a_{ij})$ and $\mathbf{A}' = (a'_{ij})$ which are chosen by the adversary $\mathcal{A}$, C computes $\mathbf{T} = \mathbf{MAN} = (t_{ij})$ and $\mathbf{T}' = \mathbf{M}'\mathbf{A}'\mathbf{N}' = (t'_{ij})$, where

$$t_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ik} \cdot a_{kl} \cdot n_{lj}$$

and

$$t'_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n} m'_{ik} \cdot a'_{kl} \cdot n'_{lj}.$$

Note that the numerical value and position of all non-zero elements of four matrices $\mathbf{M}, \mathbf{N}, \mathbf{M}'$ and $\mathbf{N}'$ are randomly chosen by C, thus t_{ij} and t'_{ij} are computationally indistinguishable. As a result, the advantage of $\mathcal{A}$ to distinguish between $\mathbf{T}$ and $\mathbf{T}'$ is negligible. $\square$

Remark 6: Currently, we cannot prove the privacy for any input $\mathbf{A}$ of **LE**. For example, if the adversary $\mathcal{A}$ chooses a nonsingular matrix $\mathbf{A}$ and a singular one $\mathbf{A}'$, then he can distinguish between $\mathbf{T} = \mathbf{MAN}$ and $\mathbf{T}' = \mathbf{M}'\mathbf{A}'\mathbf{N}'$ with an overwhelming probability since $\mathbf{T}'$ is always singular.

Even in some special case that both $\mathbf{A}$ and $\mathbf{A}'$ are nonsingular, e.g., let $\mathbf{A}$ be a nonsingular dense matrix and $\mathbf{A}'$ be the identity matrix that is extremely sparse, we do not know whether $\mathbf{T}$ and $\mathbf{T}'$ are computationally indistinguishable or not.[3] It seems that there is a paradox between the privacy and efficiency of the outsourcing scheme for any input $\mathbf{A}$. More precisely, in order to achieve privacy for any input $\mathbf{A}$, the blinding matrix $\mathbf{M}$ (and $\mathbf{N}$) for masking $\mathbf{A}$ should also be a random one (and thus may be a dense one). However, it requires $O(n^3)$ computational overhead to compute $\mathbf{T}$ (or $\mathbf{T}'$) in this case. This makes the outsourcing totally meaningless. We left it as an open problem.

Theorem 2: In the fully malicious model, the algorithms (C, S) are an $O(\frac{1}{n})$-efficient implementation of **LE**.

Proof: In the proposed algorithm **LE**, C needs to perform four matrix-vector multiplication (we omit the vector-addition operations), which takes $O(n^2)$ computations. Besides, C also needs to compute $\mathbf{T} = \mathbf{MAN}$, which also takes $O(n^2)$ computations (as discussed in **Remark 1**). On the other hand, it takes $O(n^3)$ computations in order to solve the linear equations directly. Thus, the algorithms (C, S) are an $O(\frac{1}{n})$-efficient implementation of **LE**. $\square$

Theorem 3: In the fully malicious model, the algorithms (C, S) are a 1-checkable implementation of **LE**.

Proof: Given a solution $\mathbf{y}$, C can verify whether the equations $\mathbf{Ty} = \mathbf{d}$ hold efficiently because the computational complexity for $\mathbf{Ty}$ is $O(n^2)$. Therefore, if S misbehaves

[2]The complexity of Strassen algorithm is still $O(n^{2.807})$. Coppersmith and Winograd presented the state-of-the-art record which stands at $O(n^{2.376})$. However, the programming of the two algorithms are so awkward and thus neither of them is suitable for practical applications.

[3]Note that the product of extremely sparse matrices can be complete dense [52], thus C could choose suitable $\mathbf{M}'$ and $\mathbf{N}'$ to ensure that $\mathbf{T}' = \mathbf{M}'\mathbf{N}'$ is a dense matrix. Obviously, it is impossible to distinguish $\mathbf{T}$ and $\mathbf{T}'$ only based on the sparsity of a matrix.

TABLE I
COMPARISON OF THE TWO ALGORITHMS

	Algorithm [48]	Algorithm **LE**
Computation of **ProbGen**	$(2n^2 + n)$ M $+ n^2$ EN	$((2\lambda + 1)n^2 + \lambda n)$ M
Computation of **Verify**	$(n^2 + 2L)$ M	n^2 M
Computation of **Solve**	Ln DE	λn M
Communication Round	L	1
Checkability	$1 - \frac{1}{2^l}$	1

during any execution of **LE**, it will be detected by C with probability 1. □

D. *Efficiency Analysis*

In this section, we present the theoretical analysis of efficiency for the proposed algorithm.

We denote by M an operation of multiplication, by EN an encryption (resp., by DE a decryption) of Paillier encryption systems, by L the round of iteration for algorithm in [48], by λ the maximum number of non-zero elements for each row (or column) of the chosen sparse matrices, and by l a constant. We omit other operations such as vector additions in both algorithms. Table 1 presents the comparison of the (local) computation for C, communication between C and S, and the checkability between Wang et al.'s algorithm and our proposed algorithm **LE**.[4]

Compared with Wang et al.'s algorithm, the proposed algorithm **LE** is superior in both efficiency and checkability. More precisely, **LE** requires only 1 round of communication between C and S with 100 percent checkability. Also, since $\lambda \ll n$, our algorithm **LE** is much more efficient than [48] in all phases of **ProbGen**, **Verify** and **Solve** (please also refer to the experimental results in the next section). Thus, our proposed algorithm **LE** is more suitable for real applications.

IV. Performance Evaluation

In this section, we provide a thorough experimental evaluation of the proposed outsourcing algorithm. We implement our mechanism using MATLAB language with a version of R2013a. We also utilize mathematical software MAPLE through its MATLAB interface for solving linear equations and decrypting the results. The client side process is conducted on a computer with Intel(R) Core(TM) i5-3470 CPU processor running at 3.19 GHz, 4 GB RAM. The cloud side process is conducted on a computer with Intel(R) Core(TM) i7-4770 CPU processor running at 3.40 GHz, 16 GB RAM. The test benchmark for randomly generated sparse matrices only focuses on the large-scale problems where n ranges from 5,000 to 20,000. For an extremely large n, we could use some proper matrix splitting approaches to perform large-scale matrix multiplications in case of insufficient memory. Besides, we assume that each row (or column) of the chosen sparse matrices has no more than 10 non-zero elements in our algorithm.

[4]Generally, a matrix-vector multiplication needs at most n^2 operations. However, it only requires about λn operations for a sparse-matrix-vector multiplication and $\lambda n \ll n^2$.

In order to better understand the efficiency of our proposed algorithm, we simulated all three phases (i.e., **ProbGen**, **Verify** and **Solve**) on the client side. The corresponding time costs for different size of problems are shown in Fig. 2 (we set $L = 200$). Since the time costs for different size of problems and different phases vary considerably in magnitude, we use the multiple axis breaks in Origin tools (marked with the double slash that means the vertical coordinate interrupts at this point) in the figures. The evaluation of our algorithm is based on a wide range samples. Obviously, the computation cost of client is dominated in the **ProbGen** phase that can be performed off-line, and the time cost of client is also little in the **Verify** and **Solve** phases. This implies that the most expensive computation overload is outsourced to the server in our algorithm. We will give a detailed description in the next section.

Besides, we present the comparison of computation cost for client between our algorithm **LE** and direct method in Table 2. Trivially, our algorithm **LE** can achieve noticeable computation cost savings when the dimension of the problem n increases.

A. *Cost of ProbGen*

We first consider the computation cost for customer performing **ProbGen** (i.e., the step 2 of our algorithm). As shown in Section 3.2, the main computation cost in **ProbGen** is dominated by two sparse-matrix-matrix multiplications. That is,

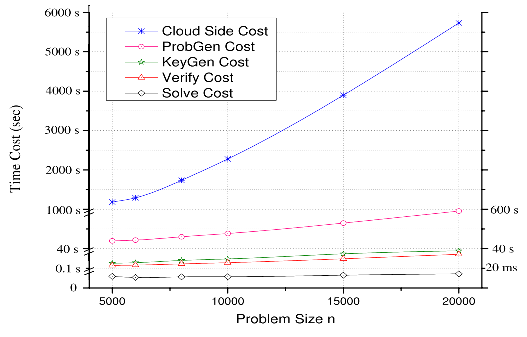
$$\mathbf{T} = \mathbf{MAN} = (\mathbf{MA})\mathbf{N}.$$

The time cost for different size of problems are shown in Table 3. We perform the experiments with n ranges from 5,000 to 20,000. For the largest benchmark size $n = 20,000$, **ProbGen** only requires less than 10 minutes on our client computer (note that this could be performed in an off-line manner). Compared to the baseline experiment where the customer solves the equation directly by himself, such computational burden should be considered practically acceptable.

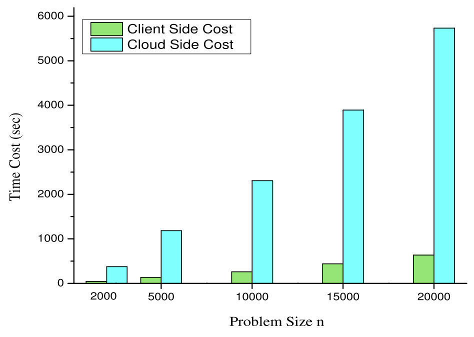
B. *Cost of Verify and Solve*

As shown in the step 4 of Section 3.2, given the computation result $\mathbf{y}$ by S, C can verify whether the equations $\mathbf{Ty} = \mathbf{d}$ hold. Hence, the main computation cost of **Verify** in our algorithm is dominated by a matrix-vector multiplication with complexity $O(n^2)$. Besides, if $\mathbf{y}$ is valid, then C can compute $\mathbf{x} = \mathbf{Ny} - \mathbf{r}$ as the solution of the original linear equations. The computation of **Solve** only requires a sparse-matrix-vector multiplication with complexity $O(n)$. Therefore, our algorithm is much superior to Wang et al.'s algorithm [48] in the phases of **Verify** and **Solve**. The time cost for different size of problems are also given in Table 3.

On the other hand, we argue that IO is used only once in our algorithm, thus it can achieve the (**optimal**) 1 round communication between C and S. However, it requires multi-round of interactive communication between C and S in [48].



(a)



(b)

Fig. 2. The efficiency comparison of LE algorithm. (a) Time cost for each phase in LE algorithm. (b) Time cost comparison between the client and cloud side.

TABLE II
COMPARISON OF COMPUTATION COST FOR CLIENT BETWEEN OUR ALGORITHM AND THE DIRECT METHOD

#	Dimension	Client Cost in **LE**	Client Cost in direct method	Speedup
1	n = 2000	44.14 sec	3769.0 sec	85.39 ×
2	n = 5000	134.38 sec	29578 sec	220.10 ×
3	n = 8000	205.44 sec	72124 sec	351.07 ×
4	n = 10000	260.96 sec	115250 sec	441.64 ×
5	n = 20000	637.50 sec	573380 sec	899.41 ×

TABLE III
STORAGE AND COMPUTATION COST OF OUR ALGORITHM FOR DIFFERENT PROBLEM SIZE

#	Dimension	Storage	**KeyGen**	**ProbGen**	**Verify**	**Solve**
1	n = 2000	32 MB	4.56 sec	37.69 sec	1.88 sec	9.42 msec
2	n = 5000	200 MB	10.14 sec	118.31 sec	5.92 sec	10.84 msec
3	n = 8000	512 MB	16.10 sec	180.31 sec	9.02 sec	11.28 msec
4	n = 10000	800 MB	18.92 sec	230.50 sec	11.53 sec	13.53 msec
5	n = 20000	3.2 GB	35.44 sec	573.38 sec	28.67 sec	16.33 msec

C. Cost of Compute (Cloud Side)

The main task of S is to solve the transformed linear equations $\mathbf{Ty} = \mathbf{d}$. It is well-known that there are two leading methods for solving large linear systems, i.e., direct methods (such as Gauss elimination method, LU decomposition method, and Cholesky decomposition method) and iterative methods (such as Jacobi iteration, Gauss-Seidel iteration and SOR iteration). Direct methods are more preferable than iterative ones in some real applications because of their robustness and predictable behavior. However, as we mentioned before, for the systems comprising hundreds of millions or even billions of equations in as many unknowns, iterative methods may be the only option available (unless the iteration does not converge). Recently, a number of efficient iterative methods were discovered and the increased need for solving huge linear systems triggered a noticeable and rapid shift toward iterative techniques in many scenarios [44].

Without loss of generality, we can take Jacobi iterative methods for example in this paper. Given the linear equations $\mathbf{Ty} = \mathbf{d}$, S first computes $\mathbf{B} = -\mathbf{D}^{-1}\mathbf{R}$ and $\mathbf{f} = \mathbf{D}^{-1}\mathbf{d}$, where $\mathbf{D}$ is the diagonal of $\mathbf{T}$, and $\mathbf{R} = \mathbf{T} - \mathbf{D}$. Then, he performs the following operations

$$\mathbf{y}^{(k+1)} = \mathbf{B}\mathbf{y}^{(k)} + \mathbf{f}$$

until it meets the requirement of given accuracy, where $\mathbf{y}^{(k)}$ is the k-th iterative result.

The time cost of cloud side for different size of problems is shown in Table 4. It is well-known that the round of iteration is related to the accuracy $\epsilon = \|\mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}\|$, the initial value $\mathbf{y}^{(0)}$, and the matrix $\mathbf{T}$. We argue that different linear equations have different round of iteration even for the same size of the problems. For example, when $n = 5000$, $\mathbf{y}^{(0)} = \{0, 0, \cdots, 0\}^T$, and $\epsilon = 10^{-6}$, the rounds of iteration vary from 32 to 340 in our experiment. This indicates that Wang et al.'s algorithm [48] may be impractical for application even for small size of n. The main reason is that the iteration in algorithm [48] requires the interactive communication between C and S (while in our algorithm C is never involved in the iteration).

D. Comparison

For the sake of completeness, we investigate the superiority of our algorithm by making an experimental comparison with algorithm [48]. To give a fair comparison, the experiments of two algorithms are both implemented on the aforementioned same computer. Recalling that algorithm [48] is a standard interactive mechanism based on Paillier's homomorphic encryption scheme, we select a 1024 bit key and set $L = 200$ likewise. Under this condition, we also take communication latency cost into consideration especially when iteration round goes larger. In our experiment, we designate communication latency as 50 msec in a scenarios with 100 Mbit/s bandwidth (for more information, please refer to [31]).

Fig. 3 illustrates the time cost of three different phases and provides a detailed comparison between two algorithms. As expected in Table 1, the experiment results shows that our algorithm is much more efficient than that in [48] for all three phases.

E. Open-Source Soft Implementation

We develop an open-source soft implementation to justify the usability of our outsourcing algorithm. The software and the source code can be downloaded as a pack from the following link: http://ste.xidian.edu.cn/cxf/le.html.

TABLE IV
CLOUD SIDE COMPUTATION COST FOR DIFFERENT SIZE OF PROBLEMS

$n = 2000$	Round of Iteration	25	59	184	369
	Time	47.11 sec	109.19 sec	351.75 sec	683.65 sec
$n = 5000$	Round of Iteration	32	56	125	340
	Time	178.95 sec	336.24 sec	720.56 sec	2007.6 sec
$n = 8000$	Round of Iteration	66	123	294	500
	Time	575.63 sec	1105.5 sec	2642.4 sec	4510.8 sec
$n = 10000$	Round of Iteration	56	88	155	300
	Time	642.30 sec	1007.7 sec	1776.4 sec	3449.6 sec
$n = 20000$	Round of Iteration	41	74	173	499
	Time	1164.8 sec	2106.3 sec	4965.7 sec	14253 sec

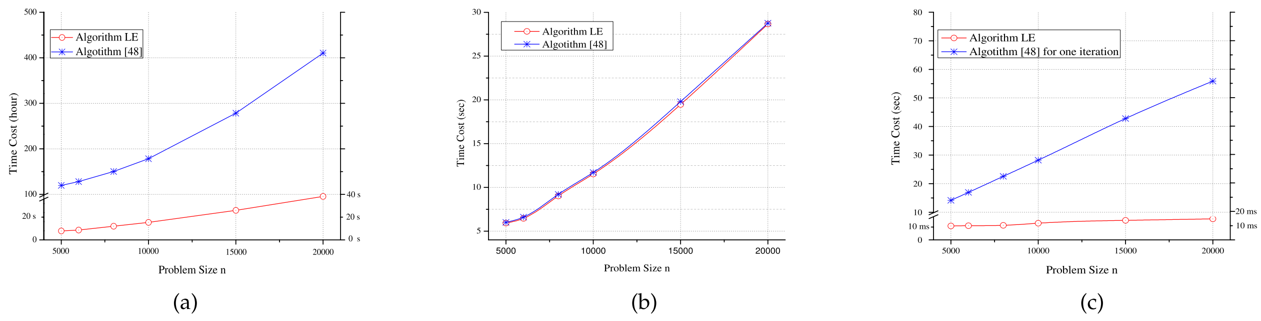


Fig. 3. The time cost comparison of three phases between two algorithms. (a) Cost of ProbGen. (b) Cost of Verify. (c) Cost of Solve.

The language we used is MATLAB with a version of R2013a on a development of Windows 32bit (MCR-R2013a(8.1)-Windows 32-bit). The developed tool is the GUI (Graphical User Interface) of MATLAB. Please firstly install the MCR installer file before using the software. The MCR installer file can be downloaded from the link: http://www.mathworks.com.

In the following, we present the guide to use the software Program.exe:

1) Open Program.exe, the interface will appear in your screen within 10 seconds.
2) Input an integer for the problem size n (e.g., 1000), and click the LeGen button in step 2. A random system of linear equations $(\mathbf{A}, \mathbf{b})$ will be generated.
3) By clicking the KeyGen button in step 3, the corresponding keys $(\mathbf{M}, \mathbf{N}, \mathbf{r})$ will be generated; Note that the length of some non-zero elements in the keys is 80-bits.
4) By clicking the ProbGen button in step 4, the values of $\mathbf{T} = \mathbf{MAN}$ and $\mathbf{d} = \mathbf{M}(\mathbf{b} + \mathbf{Ar})$ are computed.
5) By clicking the two buttons in step 5, the value $\mathbf{y}$ is computed such that $\mathbf{Ty} = \mathbf{d}$; The value $\mathbf{x}$ is computed such that $\mathbf{x} = \mathbf{Ny} - \mathbf{r}$. Trivially, $\mathbf{x}$ is the solution of the linear equations $(\mathbf{A}, \mathbf{b})$.

Remark 6: The time to run the algorithm LeGen increases rapidly with the growth of the problem size n. For example, when $n = 1000$, the time to output a random $(\mathbf{A}, \mathbf{b})$ is about 3 seconds. However, when $n = 5000$, the time to output a random $(\mathbf{A}, \mathbf{b})$ is about 2 minutes (note that LeGen will output more than 25 million random elements in this case). Also, for an extremely large n, please ensure that the computer has enough memory in order to run the software.

On the other hand, our outsourced algorithm (i.e., step 3–5 of the software Program.exe) still runs very fast even for a very large n. This further convinces the usability of our outsourcing algorithm.

V. CONCLUSIONS

In this paper, we propose an efficient outsource-secure algorithm for large-scale systems of linear equations, which are the most basic and expensive operations in many engineering disciplines. The proposed algorithm is suitable for *any* nonsingular dense matrix even in the strongest adversarial model (a.k.a. fully malicious model). Furthermore, our algorithm is superior in both efficiency and checkability than the state-of-the-art algorithm [48].

ACKNOWLEDGEMENT

We are grateful to the anonymous referees for their invaluable suggestions.

REFERENCES

[1] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2010, pp. 48–59.
[2] M. Abadi, J. Feigenbaum, and J. Kilian, "On hiding information from an oracle," in *Proc. 19th Annu. ACM Symp. Theory Comput. (STOC)*, 1987, pp. 195–203.
[3] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, "Secure outsourcing of scientific computations," *Adv. Comput.*, vol. 54, pp. 215–272, Jan. 2002.
[4] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Secur.*, vol. 4, no. 4, pp. 277–287, Oct. 2005.
[5] M. Blanton, "Improved conditional E-payments," in *Applied Cryptography and Network Security* (Lecture Notes in Computer Science), vol. 5037. Berlin, Germany: Springer-Verlag, 2008, pp. 188–206.
[6] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. 6th Annu. Conf. Privacy, Secur. Trust (PST)*, Oct. 2008, pp. 240–245.
[7] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, "Locally random reductions: Improvements and applications," *J. Cryptol.*, vol. 10, no. 1, pp. 17–36, Dec. 1997.
[8] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, Nov. 2002.
[9] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, "Multi-prover interactive proofs: How to remove intractability assumptions," in *Proc. ACM Symp. Theory Comput. (STOC)*, 1988, pp. 113–131.

[10] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 6841. Berlin, Germany: Springer-Verlag, 2011, pp. 111–131.

[11] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Computer Security* (Lecture Notes in Computer Science), vol. 7459. Berlin, Germany: Springer-Verlag, 2012, pp. 505–522.

[12] M. Blum, M. Luby, and R. Rubinfeld, "Program result checking against adaptive programs and in cryptographic settings," in *Proc. DIMACS Workshop Distrib. Comput. Crypthography*, 1990, pp. 107–118.

[13] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *J. Comput. Syst. Sci.*, vol. 47, no. 3, pp. 549–595, Dec. 1993.

[14] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1403. Berlin, Germany: Springer-Verlag, 1998, pp. 221–235.

[15] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 740. Berlin, Germany: Springer-Verlag, 1993, pp. 89–105.

[16] R. Canetti, B. Riva, and G. N. Rothblum, "Practical delegation of computation using multiple servers," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, 2011, pp. 445–454.

[17] B. Carbunar and M. Tripunitara, "Conditional payments for computing markets," in *Cryptology and Network Security* (Lecture Notes in Computer Science), vol. 5339. Berlin, Germany: Springer-Verlag, 2008, pp. 317–331.

[18] B. Carbunar and M. Tripunitara, "Fair payments for outsourced computations," in *Proc. 7th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw. (SECON)*, Jun. 2010, pp. 1–9.

[19] X. Chen, J. Li, and W. Susilo, "Efficient fair conditional payments for outsourcing computations," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 6, pp. 1687–1694, Dec. 2012.

[20] X. Chen, J. Li, J. Ma, W. Lou, and D. S. Wong, "New and efficient conditional e-payment systems with transferability," *Future Generat. Comput. Syst.*, vol. 37, pp. 252–258, Jul. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.07.015

[21] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *Computer Security* (Lecture Notes in Computer Science), vol. 7459. Berlin, Germany: Springer-Verlag, 2012, pp. 541–556.

[22] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Smart Card Research and Advanced Application* (Lecture Notes in Computer Science), vol. 6035. Berlin, Germany: Springer-Verlag, 2010, pp. 24–35.

[23] 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 Propag. Mag.*, vol. 37, no. 6, pp. 96–100, Dec. 1995.

[24] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. 20th USENIX Conf. Secur.*, 2011, p. 34. [Online]. Available: http://static.usenix.org/events/sec11/tech/full-papers/Green.pdf

[25] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 6223. Berlin, Germany: Springer-Verlag, 2010, pp. 465–482.

[26] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," in *Proc. ACM Symp. Theory Comput. (STOC)*, 2008, pp. 113–122.

[27] M. Girault and D. Lefranc, "Server-aided verification: Theory and practice," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 3788. Berlin, Germany: Springer-Verlag, 2005, pp. 605–623.

[28] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[29] P. Golle and I. Mironov, "Uncheatable distributed computations," in *Topics in Cryptology* (Lecture Notes in Computer Science), vol. 2020. Berlin, Germany: Springer-Verlag, 2001, pp. 425–440.

[30] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 19th Symp. Theory Comput.*, 1987, pp. 218–229.

[31] M. Gondree and Z. N. J. Peterson, "Geolocation of data in the cloud," in *Proc. 3rd ACM Conf. Data Appl. Secur. Privacy*, 2013, pp. 25–36.

[32] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.

[33] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 3378. Berlin, Germany: Springer-Verlag, 2005, pp. 264–282. [Online]. Available: http://www.cs.jhu.edu/~susan/papers/HL05.pdf

[34] J. Kilian, "A note on efficient zero-knowledge proofs and arguments (extended abstract)," in *Proc. ACM Symp. Theory Comput. (STOC)*, 1992, pp. 723–732.

[35] J. Kilian, "Improved efficient arguments," in *Advances in Cryptology* (Lecture Notes in Computer Science), Berlin, Germany: Springer-Verlag, 1995, pp. 311–324.

[36] A. Lathey, P. K. Atrey, and N. Joshi, "Homomorphic low pass filtering on encrypted multimedia over cloud," in *Proc. 7th IEEE Int. Conf. Semantic Comput.*, Sep. 2013, pp. 310–313.

[37] S. Micali, "CS proofs," in *Proc. 35th Annu. Symp. Found. Comput. Sci. (FOCS)*, Nov. 1994, pp. 436–453.

[38] T. Matsumoto, K. Kato, and H. Imai, "Speeding up secret computations with insecure auxiliary devices," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 403. Berlin, Germany: Springer-Verlag, 1990, pp. 497–506.

[39] M. Mohanty, W. T. Ooi, and P. K. Atrey, "Scale me, crop me, knowme not: Supporting scaling and cropping in secret image sharing," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2013, pp. 1–6.

[40] P. Q. Nguyen, I. E. Shparlinski, and J. Stern, "Distribution of modular sums and the security of server aided exponentiation," in *Proc. Workshop Comput. Number Theory Cryptograph.*, 1999, pp. 1–16.

[41] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 7194. Berlin, Germany: Springer-Verlag, 2012, pp. 422–439.

[42] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[43] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 161–174, 1991.

[44] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Boston, MA, USA: PWS Pub. Company, 1996.

[45] L. Shi, B. Carbunar, and R. Sion, "Conditional E-cash," in *Financial Cryptography and Data Security* (Lecture Notes in Computer Science), vol. 4886. Berlin, Germany: Springer-Verlag, 2007, pp. 15–28.

[46] S. Sundareswaran, A. C. Squicciarini, and D. Lin, "Ensuring distributed accountability for data sharing in the cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 4, pp. 556–568, Jul./Aug. 2012.

[47] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 6, pp. 903–916, Nov./Dec. 2012.

[48] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, Jun. 2013.

[49] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. 30th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2011, pp. 820–828.

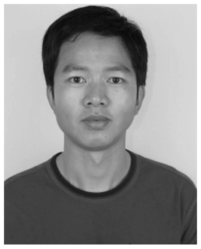
[50] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[51] W. Wu, Y. Mu, W. Susilo, and X. Huang, "Server-aided verification signatures: Definitions and new constructions," in *Provable Security* (Lecture Notes in Computer Science), vol. 5324. Berlin, Germany: Springer-Verlag, 2008, pp. 141–155.

[52] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," in *Annual European Symposium on Algorithms* (Lecture Notes in Computer Science), vol. 5324. Berlin, Germany: Springer-Verlag, 2004, pp. 604–615.

Xiaofeng Chen received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 2003, where he is currently a Professor. His research interests include applied cryptography and cloud computing security. He has authored over 100 research papers in refereed international conferences and journals. His work has been cited over 1800 times at Google Scholar. He is on the Editorial Board of *Computing and Informatics*, the *International Journal of Grid and Utility Computing*, and the *International Journal of Embedded Systems*. He has served as the Program/General Chair or a Program Committee Member for over 30 international conferences.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW, Australia, in 2009. He is currently a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, China. His research interests include cryptography and information security. He has authored over 90 research papers in refereed international conferences and journals. His work has been cited over 1500 times at Google Scholar (H-index: 22). He is on the Editorial Board of the *International Journal of Information Security* (Springer), and has served as the Program/General Chair or a Program Committee Member for over 60 international conferences.



Jin Li received the B.S. and M.S. degrees in mathematics from Southwest University, Chongqing, China, and Sun Yat-sen University, Guangzhou, China, in 2002 and 2004, respectively, and the Ph.D. degree in information security from Sun Yat-sen University, in 2007. He is currently a Professor with Guangzhou University, Guangzhou, China. His research interests include design of secure protocols in cloud computing (secure cloud storage, encrypted keyword search, and outsourcing computation) and cryptographic protocols. He served as a Senior Research Associate with the Korea Advanced Institute of Technology, Daejeon, Korea, and the Illinois Institute of Technology, Chicago, IL, USA, from 2008 to 2010, respectively. He has authored over 40 papers in international conferences and journals. He also served as a Technical Program Committee Member for many international conferences.

Jianfeng Ma received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1985, and the M.E. and Ph.D. degrees in computer software and communications engineering from Xidian University, Xi'an, China, in 1988 and 1995, respectively. From 1999 to 2001, he was with the Nanyang Technological University of Singapore, Singapore, as a Research Fellow. He is currently a Professor with the School of Computer Science, Xidian University. His current research interests include distributed systems, computer networks, and information and network security.

Wenjing Lou received the B.S. degree and the M.S. degree in computer science and engineering from Xi'an Jiaotong University, Xi'an, China, the M.A.Sc. degree in computer communications from Nanyang Technological University, Singapore, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA. She is currently an Associate Professor with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.

Duncan S. Wong received the B.Eng. (Hons.) degree in electrical and electronic engineering from the University of Hong Kong, Hong Kong, in 1994, the M.Phil. degree in information engineering from the Chinese University of Hong Kong, Hong Kong, in 1998, and the Ph.D. degree in computer science from Northeastern University, Boston, MA, USA, in 2002. He has been a Visiting Assistant Professor with the Chinese University of Hong Kong for one year before joining the City University of Hong Kong, Hong Kong, in 2003, where he is currently an Associate Professor with the Department of Computer Science.