Outsourcing Large Matrix Inversion Computation to a Public Cloud

Xinyu Lei, Xiaofeng Liao, *Senior Member*, *IEEE*, Tingwen Huang, Huaqing Li, and Chunqiang Hu

Abstract—Cloud computing enables resource-constrained clients to economically outsource their huge computation workloads to a cloud server with massive computational power. This promising computing paradigm inevitably brings in new security concerns and challenges, such as input/output privacy and result verifiability. Since matrix inversion computation (MIC) is a quite common scientific and engineering computational task, we are motivated to design a protocol to enable secure, robust cheating resistant, and efficient outsourcing of MIC to a malicious cloud in this paper. The main idea to protect the privacy is employing some transformations on the original matrix to get an encrypted matrix which is sent to the cloud, and then transforming the result returned from the cloud to get the correct inversion of the original matrix. Next, a randomized Monte Carlo verification algorithm with one-sided error is employed to successfully handle result verification. In this paper, the superiority of this novel technique in designing inexpensive result verification algorithm for secure outsourcing is well demonstrated. We analytically show that the proposed protocol simultaneously fulfills the goals of correctness, security, robust cheating resistance, and high efficiency. Extensive theoretical analysis and experimental evaluation also show its high efficiency and immediate practicability.

Index Terms—Cloud computing, matrix inversion, secure outsourcing, robust cheating resistant, Monte Carlo verification

1 Introduction

With the emergence of the cloud computing paradigm in scientific and business applications, it has become increasingly important to provide service-oriented computing in third-party data management settings. With this paradigm, the resource-constrained clients can offload their intensive computational tasks to clouds, which are equipped with massive computational resources. In contrast to setting up and maintaining their own infrastructures, the clients can economically share the massive computational power, storage, and even some softwares of the clouds.

1.1 Challenges

Promising as it is, outsourcing computational problem to the commercial public service provider inevitably brings in new security concerns and challenges [1]. The first challenge is the client's input/output data privacy. The outsourced computational problems and the results to these problems often contain sensitive information. To hide the sensitive information from the cloud, the client needs to encrypt their data before outsourcing and decrypt the returned result from the cloud after outsourcing. The second challenge is the verification of the result returned by the cloud. This is because the cloud may behave unfaithfully and return incorrect results. As an example of intentional reason, for the outsourced computational intensive tasks, there are strong financial incentives for the cloud to be lazy and just return incorrect answers to the client if such answers require less work and are unlikely to be detected by the client. Besides, some accidental reasons such as possible software bugs or hardware failures may also result in a wrong computing. Consequently, the outsourcing protocol must be designed in such a way that it is able to detect whether the returned result is correct or not. The third challenge is efficiency. On one hand, a key requirement is that the amount of local work performed by the client must be substantially cheaper than performing the original computational problem on its own. Otherwise, there is no point for the client to resort to the cloud. On the other hand, it is also desirable to keep the amount of work performed by the cloud as close as possible to that needed to compute the original problem by the client itself. Otherwise, the cloud may be unable to complete the task in a reasonable amount of time, or the cost of the cloud may become prohibitive. To sum up, an outsourcing computation protocol should satisfy four aspects: it is correct, secure, verifiable, and efficient.

1.2 Motivations

Matrix inversion computation (MIC) is a basic computational problem in scientific and engineering fields and has a number of applications. For example, MIC plays a significant role in scientific computations. Take a typical linear regression model $\mathbf{y} = \mathbf{X}\beta$ as an example, the least squared error method yields a solution for β by computing $\beta = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$ [2]. Besides, MIC is widely used in computer graphics, particularly in 3D graphics rendering and 3D simulations [3]. Examples include screen-to-world ray casting, world-to-subspace-to-world object transformations,

- *X. Lei, X. Liao, and H. Li are with the State Key Laboratory of Power Transmission Equipment & System Security and New Technology, College of Computer Science, Chongqing University, Chongqing 400044, P.R. China. E-mail: xy-lei@qq.com, xfliao@cqu.edu.cn, lhq_jsack@126.com.*
- *T. Huang is with the Texas A&M University at Qatar, Doha, PO Box 23874, Qatar. E-mail: tingwen.huang@qatar.tamu.edu.*
- *C. Hu is with the Department of Computer Science, George Washington University, Washington, DC 20052. E-mail: chu@gwu.edu.*

Manuscript received 24 Mar. 2013; revised 3 Aug. 2013; accepted 18 Sept. 2013; published online 26 Sept. 2013.
Recommended for acceptance by D. Lie.
For information on obtaining reprints of this article, please send e-mail to: tcc@computer.org, and reference IEEECS Log Number TCC-2013-03-0057.
Digital Object Identifier no. 10.1109/TCC.2013.7.

2168-7161/13/$31.00 © 2013 IEEE Published by the IEEE CS, ComSoc, PES, CES, & SEN

and physical simulations. Moreover, MIC is well rooted is many other engineering and scientific fields including image encryption [4], [5], image watermarking [6], [7], to just list a few. To sum up, MIC is widely needed for a variety of clients. When the restricted computational resources are possessed by these clients and MIC deals with a large matrix (or a batch of large matrices), an economical solution is to outsource MIC to a powerful cloud. Even if the data are in a moderate scale, for clients as battery-limited mobile phones, portable devices, or embedded smart cards, secure outsourcing of MIC is preferred. Consequently, we are motivated to design a protocol to enable clients to securely, verifiably, and efficiently outsource MIC to a cloud.

1.3 Main Contributions

This paper addresses the issue of how to outsource MIC to a remote malicious cloud server while ensuring correctness, maintaining data input/output privacy, realizing result verifiability, and improving computational efficiency. From the complexity point of view, matrix multiplication and matrix inversion of square matrices are essentially the same problem. The hitherto best algorithm known for the latter resorts to that of the former. Most efficient algorithms to both problems hence share the same time complexity of $O(n^{2.373})$. An important challenge in encrypting the input matrix $\mathbf{X}$ is, therefore, to avoid multiplying the original matrix $\mathbf{X}$ with general matrices, for avoiding a complexity that is the same as inverting $\mathbf{X}$ itself. By applying permutation functions, this paper describes a way of multiplying $\mathbf{X}$ with special matrices, where matrix product can be computed in $O(n^2)$ time. Besides, the challenge in the result verification step is also to avoid general matrix multiplication, since the validity of a returned matrix can be easily checked by taking a product of that matrix with the original input $\mathbf{X}$, and check whether an identity matrix is obtained. By introducing Monte Carlo verification algorithm, the proposed protocol is able to verify the correctness of the returned result in $O(n^2)$ time. Based on the permutation technique and Monte Carlo technique, the client can reduce its original $O(n^{2.373})$ work to $O(n^2)$ work by outsourcing MIC to a cloud. Moreover, experimental evaluation is also provided to show that the proposed protocol is able to allow the client to outsource MIC to a cloud and gain substantial computation savings.

1.4 Organization

The remainder of this paper proceeds as follows: Section 2 introduces some essential preliminaries. In Section 3, we describe our protocol with detailed techniques. Sections 4 and 5 give some related analysis and performance evaluation, followed by Section 6 which overviews the related work. Finally, some conclusions are drawn in Section 7.

2 Preliminaries

2.1 System Model, Threat Model, Design Goals, and Framework

2.1.1 System Model

We consider the secure MIC outsourcing system model, as illustrated in Fig. 1. A client with low computational power wants to outsource the original MIC to a cloud service provider, who has massive computational power and special software. To protect input privacy, the client encrypts the original MIC using a secret key K to get an MIC problem, written as MIC_K. Later, the encrypted MIC_K is given to the cloud for a result. Once the cloud receives MIC_K, the computation is carried out with software; then, the cloud sends back the result to MIC_K. The cloud also sends back a proof Γ that tries to prove the returned result is indeed correct and the cloud does not cheat. On receiving the returned result, the client decrypts the returned result using the secret key K to get the result to the original MIC. Meanwhile, the client checks whether this result is correct: if yes, accepts it; if no, just rejects it.

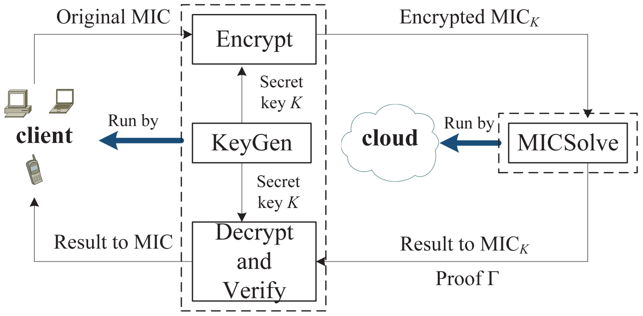


Fig. 1. Secure MIC outsourcing system model.

2.1.2 Threat Model

The security threats faced by the outsourcing system model primarily come from the behavior of the cloud. Generally, there are two levels of threat models in outsourcing: semihonest cloud model and malicious cloud model [8]. In the semihonest cloud model, the cloud correctly follows the protocol specification. However, the cloud records all the information it can access and attempts to use this to learn information that should remain private. While in the malicious cloud model, the cloud can arbitrarily deviate from the protocol specification. The malicious cloud may just return a random result to the client to save its computing resources, while hoping not to be detected by the client. Therefore, an outsourcing protocol in the malicious cloud model should be able to handle result verification. In this paper, we assume that the cloud is malicious. The proposed protocol should be able to resist such a malicious cloud.

2.1.3 Design Goals

We identify the following goals that the outsourcing protocol should satisfy.

- *Correctness.* If both the client and the cloud follow the protocol honestly, the MIC can be indeed fulfilled by the cloud and the client gets a correct result to the original MIC.
- *Security.* The protocol can protect the privacy of the client's data. On one hand, given the encrypted MIC_K problem, the cloud cannot get meaningful knowledge of the client's input data, which are referred to as *input privacy*. On the other hand, the correct result to the original MIC is also hidden from the cloud, and this is called as *output privacy*.
- *Robust cheating resistance.* The correct result from a faithful cloud server must be verified successfully by

the client. No false result from a cheating cloud server can pass the verification with a non-negligible probability.

- *Efficiency.* The local computation done by the client should be substantially less than the computation of the original MIC on his own. In addition, the amount of computation on computing the encrypted MIC_K should be as close as possible to that on computing the original MIC.

2.1.4 Framework

Syntactically, a secure MIC outsourcing protocol should contain five subalgorithms:

1. the algorithm for key generation KeyGen,
2. the algorithm for MIC encryption MICEnc,
3. the algorithm for solving MIC_K problem MICSolve,
4. the algorithm for MIC decryption MICDec, and
5. the algorithm for result verification ResultVerify.

One significant difference between this framework and the traditional encryption framework is that in this case, both encryption and decryption processes occur in the client side. This eliminates the expensive public key exchange process in the traditional encryption framework. Therefore, this framework is able to efficiently realize one-time-pad type of flexibility. That is to say, KeyGen will be run every time for a new outsourced matrix instance to enhance security. Once we have this framework, we just need to work out the details of these five sub-algorithms, which will be shown in Section 3.

2.2 Mathematical Background

Permutation function is well studied in group theory and combinatorics. In Cauchy's two-line notation, one lists the preimage element in the first row and, for each preimage element, lists its image under the permutation below it in the second row. Then, the permutation function can be written as

$$\begin{pmatrix} 1 & \cdots & n \\ p_1 & \cdots & p_n \end{pmatrix}. \tag{1}$$

We use a permutation function $\pi(i) = p_i$, where $i = 1, \ldots, n$, to denote (1). Let π^{-1} denote the inverse function of π. The Kronecker delta function $\delta_{x,y}$ is defined as

$$\delta_{x,y} = \begin{cases} 1, & x = y, \\ 0, & x \neq y. \end{cases} \tag{2}$$

Let $\mathbf{I}$ be an $n \times n$ identity matrix and $\mathbf{0}$ be an $n \times n$ zero matrix. For a matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$, let $\mathbf{X}(i,j)$, $x_{i,j}$, or x_{ij} denote the entry in ith row and jth column in matrix $\mathbf{X}$, where i and j are indexed from 1 to n.

3 Protocol Construction

In this section, each part of the framework for secure outsourcing of MIC will be individually solved.

3.1 Secret Key Generation

Consider a nonsingular matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$; the resource-constrained client wants to securely outsource the computation of $\mathbf{X}^{-1}$ to the cloud. The protocol starts by invoking Procedure Secret-Key-Generation to set up a secret key K.

Algorithm 1. Procedure Secret-Key-Generation.
Input: A security parameter λ.
Output: Secret key K: $\{\alpha_1, \ldots, \alpha_n\}$, $\{\beta_1, \ldots, \beta_n\}$, π_1, π_2.

```
On input a security parameter $\lambda$, which specifies key
  space $\mathcal{K}_\alpha$ and $\mathcal{K}_\beta$, the client picks two sets of random
  numbers: $\{\alpha_1, \ldots, \alpha_n\} \leftarrow \mathcal{K}_\alpha$, $\{\beta_1, \ldots, \beta_n\} \leftarrow \mathcal{K}_\beta$, where
  $0 \notin \mathcal{K}_\alpha \cup \mathcal{K}_\beta$.
The client invokes Algorithm 2 to generate two random
  permutations $\pi_1$ and $\pi_2$ of the integers $1, \ldots, n$.
```

Algorithm 2. Random Permutation Generation.

```
Set $\pi = \mathrm{I}_n$. (identical permutation)
for $i = n$ down to 2
    Set $j$ to be a random integer with $1 \leq j \leq i$.
    Swap $\pi[j]$ and $\pi[i]$.
end for
```

Algorithm 2 is due to Durstenfeld [9]; it is usually called Fisher-Yates shuffle [10]. There are several variants of Algorithm 2 to generate a random permutation. Nevertheless, the asymptotic time complexity of Algorithm 2 has already been optimal. This is the reason for this algorithm to be used for random permutation generation in this work.

3.2 MIC Encryption.

Next, we describe Procedure MIC-Encryption.

Algorithm 3. Procedure MIC-Encryption.
Input: The original matrix $\mathbf{X}$ and the Secret key K: $\{\alpha_1, \ldots, \alpha_n\}$, $\{\beta_1, \ldots, \beta_n\}$, π_1, π_2.
Output: $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$.

```
The client generates matrices $\mathbf{P}_1$, $\mathbf{P}_2$, where
  $\mathbf{P}_1(i,j) = \alpha_i\delta_{\pi_1(i),j}$, $\mathbf{P}_2(i,j) = \beta_i\delta_{\pi_2(i),j}$.
The client computes $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$. According to
  Theorem 1, the client can use (4) to efficiently (via time
  $O(n^2)$) compute $\mathbf{Y}$.
Later, the encrypted matrix $\mathbf{Y}$ will be outsourced to
  the cloud.
```

Lemma 1. *In Procedure MIC-Encryption, matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ are invertible. More precisely,*

$$\begin{cases} \mathbf{P}_1^{-1}(i,j) = (\alpha_j)^{-1}\delta_{\pi_1^{-1}(i),j}, \\ \mathbf{P}_2^{-1}(i,j) = (\beta_j)^{-1}\delta_{\pi_2^{-1}(i),j}. \end{cases} \tag{3}$$

Proof. Since $0 \notin \mathcal{K}_\alpha \cup \mathcal{K}_\beta$, the determinants of $\mathbf{P}_1$, $\mathbf{P}_2$ satisfy $\det(\mathbf{P}_1) \neq 0$, $\det(\mathbf{P}_2) \neq 0$. Hence, $\mathbf{P}_1$ and $\mathbf{P}_2$ are invertible. Hereafter, the proof is straightforward. □

Theorem 1. *In Procedure MIC-Encryption, if $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$, then it holds that*

$$\mathbf{Y}(i,j) = (\alpha_i/\beta_j)\mathbf{X}(\pi_1(i), \pi_2(j)). \tag{4}$$

Proof. Let

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \ldots & x_{n,n} \end{bmatrix}. \tag{5}$$

Observe that $\mathbf{P}_1(i,j) = \alpha_i \delta_{\pi_1(i),j}$, this leads to

$$\mathbf{P}_1\mathbf{X} = \begin{bmatrix} \alpha_1 x_{\pi_1(1),1} & \cdots & \alpha_1 x_{\pi_1(1),n} \\ \vdots & \ddots & \vdots \\ \alpha_i x_{\pi_1(i),1} & \cdots & \alpha_i x_{\pi_1(i),n} \\ \vdots & \ddots & \vdots \\ \alpha_n x_{\pi_1(n),1} & \cdots & \alpha_n x_{\pi_1(n),n} \end{bmatrix}. \tag{6}$$

It holds from Lemma 1 that $\mathbf{P}_2^{-1}(i,j) = (\beta_j)^{-1}\delta_{\pi_2^{-1}(i),j}$. Then, one can obtain

$$\mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1} = \begin{bmatrix} \frac{\alpha_1}{\beta_1} x_{\pi_1(1),\pi_2(1)} & \cdots & \frac{\alpha_1}{\beta_j} x_{\pi_1(1),\pi_2(j)} & \cdots & \frac{\alpha_1}{\beta_n} x_{\pi_1(1),\pi_2(n)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\alpha_i}{\beta_1} x_{\pi_1(i),\pi_2(1)} & \cdots & \frac{\alpha_i}{\beta_j} x_{\pi_1(i),\pi_2(j)} & \cdots & \frac{\alpha_i}{\beta_n} x_{\pi_1(i),\pi_2(n)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\alpha_n}{\beta_1} x_{\pi_1(n),\pi_2(1)} & \cdots & \frac{\alpha_n}{\beta_j} x_{\pi_1(n),\pi_2(j)} & \cdots & \frac{\alpha_n}{\beta_n} x_{\pi_1(n),\pi_2(n)} \end{bmatrix}. \tag{7}$$

This can be finally rewritten as $\mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1} = \mathbf{Y}(i,j) = (\alpha_i/\beta_j)\mathbf{X}(\pi_1(i), \pi_2(j))$, completing the proof. □

3.3 MIC in the Cloud

See Procedure MIC_K-in-the-Cloud.

Algorithm 4. Procedure MIC_K-in-the-Cloud.
Input: $\mathbf{Y}$.
Output: $\mathbf{R}' = \mathbf{Y}^{-1}$.

```
1: On input the encrypted matrix $\mathbf{Y}$, the cloud then
   invokes any matrix inversion algorithm to compute
   $\mathbf{R}' = \mathbf{Y}^{-1}$.
2: The cloud then sends matrix $\mathbf{R}'$ back to the client.
```

3.4 MIC Decryption

See Procedure MIC-Decryption.

Algorithm 5. Procedure MIC-Decryption.
Input: $\mathbf{R}'$ and K.
Output: $\mathbf{R}$.

```
1: On receiving the returned matrix $\mathbf{R}'$ from the cloud, the
   client compute $\mathbf{R} = \mathbf{P}_2^{-1}\mathbf{R}'\mathbf{P}_1$. According to Theorem 2,
   the client can use (8) to efficiently (via time $O(n^2)$)
   compute $\mathbf{R}$.
```

Theorem 2. *In Procedure Result-Verification, if* $\mathbf{R} = \mathbf{P}_2^{-1}\mathbf{R}'\mathbf{P}_1$, *then it holds that*

$$\mathbf{R}(i,j) = \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(i)}\right)\mathbf{R}'(\pi_2^{-1}(i), \pi_1^{-1}(j)). \tag{8}$$

Proof. The detailed proof is similar to Theorem 1. We now briefly describe the proof. By Lemma 1, we have $\mathbf{P}_2^{-1}(i,j) = (\beta_j)^{-1}\delta_{\pi_2^{-1}(i),j}$. Together with $\mathbf{P}_1(i,j) = \alpha_i\delta_{\pi_1(i),j}$, then (8) can be deduced from $\mathbf{R} = \mathbf{P}_2^{-1}\mathbf{R}'\mathbf{P}_1$. □

3.5 Result Verification

Generally, handling result verification is not an easy task. However, this problem is well addressed by using the idea of Freivalds' algorithm [11], [12]. Technique details are elaborated in Procedure Result-Verification. We defer the detailed analysis of it in Section 4.

Algorithm 6. Procedure Result-Verification.
Input: The decrypted but unchecked result $\mathbf{R}$.
Output: Accepts $\mathbf{R}$ as a correct result; or rejects it.

```
1: for $i = 1 : l$ /*parameter $l$ represents the running times
   of the random check process from Step 2 to Step 6*/
2:     The client selects an $n \times 1$ random 0/1 vector $\mathbf{r}$.
3:     The client computes $\mathbf{P} = \mathbf{R} \times (\mathbf{X}\mathbf{r}) - \mathbf{I} \times \mathbf{r}$.
4:     if $\mathbf{P} \neq (0, \ldots, 0)^{\mathrm{T}}$
5:         Output "verification fails;" aborts.
6:     end if
7: end for
8: The client accepts $\mathbf{R}$ as a correct result if it passes
   the above check; otherwise, rejects it.
```

3.6 The Completed Protocol

We now present the completed protocol that contains five subalgorithms (KeyGen, MICEnc, MICSolve, MICDec, ResultVerify) as follows:

- KeyGen(1^λ). On input a security parameter λ, the client invokes Procedure Secret-Key-Generation to get a secret key K: $\{\alpha_1, \ldots, \alpha_n\}$, $\{\beta_1, \ldots, \beta_n\}$, π_1, π_2.
- MICEnc($\mathbf{X}; K$). On input the original matrix $\mathbf{X}$ and the secret key K, the client invokes Procedure MIC-Encryption to encrypt the original matrix $\mathbf{X}$ into an encrypted matrix $\mathbf{Y}$ to protect the input privacy.
- MICSolve($\mathbf{Y}$). On input the encrypted matrix $\mathbf{Y}$, the cloud invokes Procedure MIC_K-in-the-Cloud to get a result $\mathbf{R}'$. Then, the cloud returns $\mathbf{R}'$ and an empty proof Γ to the client.
- MICDec($\mathbf{R}', K$). On input the returned result $\mathbf{R}'$ and the secret key K, the client invokes Procedure MIC-Decryption to get an unchecked result $\mathbf{R}$.
- ResultVerify($\mathbf{R}, \Gamma$). On input the unchecked result $\mathbf{R}$ and the empty proof Γ, the client invokes Procedure Result-Verification to check its correctness. If it passes the check, then accepts $\mathbf{R}$ as the correct inversion of the original matrix $\mathbf{X}$; otherwise, just rejects it.

4 Correctness, Security, and Verifiability Analysis

4.1 Correctness Guarantee

Theorem 3. *The proposed protocol is correct.*

Proof. It suffices to show that if both the client and the cloud follow the protocol honestly, the corresponding decrypted result $\mathbf{R}$ is always the correct inversion of the original matrix $\mathbf{X}$.

The client first computes $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$. Next, an honest cloud server computes

$$\mathbf{R}' = \mathbf{Y}^{-1} = \mathbf{P}_2\mathbf{X}^{-1}\mathbf{P}_1^{-1}. \tag{9}$$

Then, in Procedure MIC-Decryption, the client computes

$$\mathbf{R} = \mathbf{P}_2^{-1}\mathbf{R}'\mathbf{P}_1 = \mathbf{X}^{-1}. \tag{10}$$

This implies the proposed protocol is correct. □

4.2 Security Guarantee

Input Privacy. The proposed protocol can protect input privacy if the cloud cannot recover the original matrix $\mathbf{X}$ from the encrypted matrix $\mathbf{Y}$. The original matrix $\mathbf{X}$ is encrypted by the following two phases:

- *Phase 1*. The position of each entry in the original matrix is randomly rearranged under two random permutations, i.e., $\mathbf{T}(i,j) = \mathbf{X}(\pi_1(i), \pi_2(j))$.
- *Phase 2*. Each entry in matrix $\mathbf{T}$ is further masked by multiplying a factor, i.e., $\mathbf{Y}(i,j) = (\alpha_i/\beta_j)\mathbf{T}(i,j)$.

In Phase 1, the key space consists of all random permutations of π_1, π_2, meaning that there are $(n!)^2$ cases of permutations. Each case occurs with probability $\frac{1}{(n!)^2}$. This implies that even if the cloud has the correct matrix $\mathbf{T}$, the expected time of brute-force attack on the key space to recover the original matrix $\mathbf{X}$ is $\frac{(n!)^2}{2}$, which is definitely a nonpolynomially bounded quantity in terms of n. In Phase 2, each entry in matrix $\mathbf{T}$ is further masked, the expected time of brute-force attack on the key space to guess $\{\alpha_1, \ldots, \alpha_n\}$ and $\{\beta_1, \ldots, \beta_n\}$ is $\frac{|\mathcal{K}_\alpha|^n|\mathcal{K}_\beta|^n}{2}$. A choice of large key space $\mathcal{K}_\alpha$ and $\mathcal{K}_\beta$ will thwart this attack. Consider the above, without the each-run-specific secret key, the cloud cannot recover $\mathbf{X}$ from $\mathbf{Y}$ by trivial means. In this way, input privacy is protected.

Output Privacy. The proposed protocol can protect output privacy if the cloud cannot recover the correct inversion $\mathbf{R}$ from the computational result $\mathbf{R}'$. According to (8), it is evident that output privacy is protected in the same way as input privacy. The detailed analysis is omitted accordingly.

Remarkably, the security analysis follows an informal approach. A meaningful and challenging future work lies in giving a rigorous proof of security.

4.3 Verifiability Guarantee

Theorem 4. *The proposed protocol satisfies robust cheating resistance.*

Proof. The correctness of the decrypted result $\mathbf{R}$ is checked from Step 1 to Step 7 in Procedure Result-Verification. The random check process from Step 2 to 6 is repeated l times. We first define the following two parameters to facilitate our proof. Let $Prob_1$ be the probability of *a false negative* (nondetection of a false returned result) in one round of a random check process. Let $Prob_f$ denote the probability of *check failure*, i.e., the probability of occurrence of a false negative in whole l times random check processes.

The proof consists of two steps. First, we show that the result from a faithful cloud server must be verified successfully by the client. From Theorem 3, if the cloud is faithful, we have $\mathbf{R} = \mathbf{X}^{-1}$. This leads to $\mathbf{R} \times \mathbf{X} = \mathbf{I}$. So

$$\mathbf{P} = \mathbf{R} \times (\mathbf{X}\mathbf{r}) - \mathbf{I} \times \mathbf{r} = (0, \ldots, 0)^{\mathrm{T}}, \tag{11}$$

regardless of what vector $\mathbf{r}$ is. In such case, the verification failure step (Step 5 in Procedure Result-Verification) will never be executed. This means that a correct result $\mathbf{R}$ must be verified successfully by the client.

Next, we show that no false result from a cheating cloud server can pass the verification with a non-negligible probability. In other words, we attempt to prove that $Prob_f$ is a negligible quantity. Let

$$\mathbf{D} = \mathbf{R} \times \mathbf{X} - \mathbf{I}, \mathbf{P} = \mathbf{D} \times \mathbf{r} = (p_1, \ldots, p_n)^{\mathrm{T}}. \tag{12}$$

If the cheating cloud return a false $\mathbf{R}'$, then this leads to $\mathbf{R} \neq \mathbf{X}^{-1}$. Accordingly, we have $\mathbf{R} \times \mathbf{X} - \mathbf{I} \neq \mathbf{0}$, so at least one element of $\mathbf{D}$ is nonzero. Suppose that the element $d_{ij} \neq 0$. By the definition of matrix-vector multiplication, we obtain

$$p_i = \sum_{k=1}^{n} d_{ik} r_k = d_{i1} r_1 + \cdots + d_{ij} r_j + \cdots + d_{in} r_n = d_{ij} r_j + y, \tag{13}$$

where $y = \sum_{k=1}^{n} d_{ik} r_k - d_{ij} r_j$. Applying Total Probability Theorem, it holds that

$$\begin{aligned} \Pr[p_i = 0] &= \Pr[p_i = 0 | y = 0]\Pr[y = 0] \\ &\quad + \Pr[p_i = 0 | y \neq 0]\Pr[y \neq 0]. \end{aligned} \tag{14}$$

Note from (13) that

$$\begin{cases} \Pr[p_i = 0 | y = 0] = \Pr[r_j = 0] = 1/2, \\ \Pr[p_i = 0 | y \neq 0] \leq \Pr[r_j = 1] = 1/2. \end{cases} \tag{15}$$

Substituting (15) into (14) results in

$$\Pr[p_i = 0] \leq (1/2)\Pr[y = 0] + (1/2)\Pr[y \neq 0]. \tag{16}$$

Putting $\Pr[y \neq 0] = 1 - \Pr[y = 0]$ into (16) leads to

$$\Pr[p_i = 0] \leq 1/2. \tag{17}$$

Based on (17), $Prob_1$ satisfies

$$Prob_1 = \Pr[\mathbf{P} = (0, \ldots, 0)^{\mathrm{T}}] \leq \Pr[p_i = 0] \leq \frac{1}{2}. \tag{18}$$

Observe that the random check process is repeated l times, $Prob_f$ can be estimated by

$$Prob_f \leq Prob_1^l \leq \frac{1}{2^l}, \tag{19}$$

which demonstrates that $Prob_f$ is a negligible quantity in terms of l. The proof is completed. □

It can be deduced from the proof of Theorem 4 that the proposed protocol can handle result verification with check failure probability at most 2^{-l}. The size of l is a tradeoff between cheating resistance and efficiency. A conservative choice of high cheating resistance should require l to be around 80 bits (in this case $Prob_f \leq \frac{1}{2^{80}}$). For a fast check, a reasonable choice of 20 bits is also acceptable (in this case $Prob_f \leq \frac{1}{2^{20}}$). A similar case of tradeoff can be found in [13].

4.4 Further Discussions on Result Verification

MONTE CARLO VERIFICATION ALGORITHM. Let us proceed to introduce the notion of Monte Carlo verification algorithm, which is formally defined below.

Definition 1 (Monte Carlo Verification Algorithm [12]). *The classification and definition of Monte Carlo verification algorithm are summarized in Table 1. The detailed verbal description of case 1 is as follows: for a randomized verification algorithm Vrfy and any decrypted but unchecked result Res, if*

TABLE 1
Classification and Definition of Monte Carlo Verification Algorithm

Cases	Satisfied conditions	Definitions
case 1	Pr[Vrfy accepts $Res\|Res$ is correct] $=1$, Pr[Vrfy accepts $Res\|Res$ is false] $\leq \delta$.	True-biased Monte Carlo verification algorithm with one-sided error δ
case 2	Pr[Vrfy accepts $Res\|Res$ is correct] $\geq \epsilon$, Pr[Vrfy accepts $Res\|Res$ is false] $=0$.	False-biased Monte Carlo verification algorithm with one-sided error ϵ
case 3	Pr[Vrfy accepts $Res\|Res$ is correct] $\geq \epsilon$, Pr[Vrfy accepts $Res\|Res$ is false] $\leq \delta$.	Monte Carlo verification algorithm with two-sided errors (δ, ϵ)

$$\begin{aligned}\Pr[\text{Vrfy accepts } Res|Res \text{ is correct }] &= 1,\\ \Pr[\text{Vrfy accepts } Res|Res \text{ is false }] &\leq \delta,\end{aligned} \tag{20}$$

then we define Vrfy as a true-biased Monte Carlo verification algorithm with one-sided error δ. The detailed verbal description of cases 2 and 3 can be analogously obtained.

Based on Definition 1 and the proof of Theorem 4, we immediately have the following theorem.

Theorem 5. *One round of random check process, i.e., from Step 2 to Step 6 in Procedure Result-Verification, is a true-biased Monte Carlo verification algorithm with one-sided error* $\frac{1}{2}$.

For a Monte Carlo verification algorithm with one-sided error, the failure probability can be reduced (or the success probability amplified) by running the algorithm multiple times. Indeed, this mechanism has been exploited in Procedure Result-Verification. According to the above analysis, case 2 and case 3 of Monte Carlo verification algorithms (see Table 1) can also be used in designing practical result verification algorithms for secure outsourcing. One may see in what follows that Monte Carlo verification algorithm offers superiority in designing efficient result verification algorithm, which is generally a difficult task in secure outsourcing.

5 Performance Evaluation

5.1 Theoretical Results

Client-Side Overhead. The client-side overhead is generated by running four subalgorithms: KeyGen, MICEnc, MICDec, and ResultVerify. It is evident that KeyGen takes time $O(n)$. In MICEnc, applying (4) to efficiently compute $\mathbf{Y}$, it only takes time $O(n^2)$. Likewise, the time consumed by MICDec is $O(n^2)$. As to ResultVerify, the time is dominated by computing $\mathbf{R} \times (\mathbf{X}\mathbf{r})$, which takes time $O(n^2)$.

Cloud-Side Overhead. For the cloud, its only computation overhead is generated by running MICSolve. The cloud can apply any existing matrix inversion algorithm. As mentioned before, from the complexity point of view, matrix multiplication and matrix inversion of square matrices are essentially the same problem. Table 2 shows representative algorithms for MIC. From Table 2, the time consumed by MICSolve is $O(n^\rho)(2.373 \leq \rho \leq 3)$.

Shown in Table 3 is a summarization of the theoretical results. More specifically, the overall time cost is $O(n^2)$ for the client and $O(n^\rho)$ for the cloud. From the perspective of efficiency, the proposed protocol is feasible because there exists a gap between $O(n^2)$ and $O(n^\rho)$. According to the asymptotic behavior of Big-O notation [17], we have that the computational overhead in the client side will be less than that in the cloud side for a sufficiently large n. The theoretical results indicate that the proposed protocol is able to allow the client to outsource MIC to the cloud and gain substantial computation savings. This claim will be further validated by our experiments in the next section.

5.2 Experimental Results

Theoretical analysis of the protocol has shown that outsourcing indeed benefits the client. We proceed to implement the protocol to assess its practical efficiency in this section. Both client and cloud server computations in our experiments are conducted on a same workstation. If we implement the protocol for both client side and cloud side on a same workstation and measure their running time, then the ratio of time (see the definition of cloud efficiency in the next paragraph) can reflect the asymmetric amount of computation performed in both sides. However, if we implement the protocol on two different workstations with one being client and the other being cloud server, then the cloud efficiency will be case-specific, depending on the asymmetric computing speed owned by the two different workstations. Consequently, one-workstation-based experiment is employed. In reality, the cloud server is always more powerful and this will further reduce the running time measured in our experiment. Besides, we ignore the communication latency between the client and the cloud for this application because the computation dominates the running time as shown in our experiments.

Our goal is to find the performance gain for the client by outsourcing. Thus, the main performance indicator is a ratio of the time that is needed if the computation is done locally over the time that is needed by the client's computation if outsourcing is chosen. With clear definition of parameters in Table 4, the performance gain of the client can be shown by $\frac{t_{\text{original}}}{t_{\text{client}}}$, and we refer to this as *client speedup*. This value theoretically should be a considerable positive number greater than 1, which means there is a considerable performance gain. We also consider another metric, i.e., the *cloud efficiency*, using $\frac{t_{\text{original}}}{t_{\text{cloud}}}$. Ideally, the MIC encryption

TABLE 2
Representative Algorithms for Matrix Inversion

Algorithms	Time complexity
Gauss-Jordan elimination	$O(n^3)$
Strassen algorithm [14]	$O(n^{2.807})$
Coppersmith-Winograd algorithm [15]	$O(n^{2.376})$
Williams algorithm [16]	$O(n^{2.373})$

TABLE 3
Theoretical Performance of the Proposed Protocol

Client					Cloud	
KeyGen	MICEnc	MICDec	ResultVerify	Sending cost	MICSolve	Sending cost
$O(n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	Matrix $\mathbf{Y}$	$O(n^\rho)(2.373 \leq \rho \leq 3)$	Matrix $\mathbf{R}'$

TABLE 4
Notations

Notations	Means
t_{original}	the time for the client to compute the original MIC locally
t_{cloud}	the time for the cloud to compute the outsourced MIC_K
t_{client1}	the time for the client to generate the secret key and encrypt the original MIC
t_{client2}	the time for the client to decrypt and verify the returned result
t_{client}	$t_{\text{client}} = t_{\text{client1}} + t_{\text{client2}}$

TABLE 5
Experimental Performance of the Proposed Protocol (Time in Seconds)

$l = 20$ with check failure probability $Prob_f \leq \frac{1}{2^{20}}$: **Low Cheating Resistance and High Client Speedup**

Benchmark		Original MIC	Encrypted MIC_K				Client Speedup	Cloud Efficiency
No.	dimension n	t_{original}	t_{cloud}	t_{client1}	t_{client2}	t_{client}	$t_{\text{original}}/t_{\text{client}}$	$t_{\text{original}}/t_{\text{cloud}}$
1	250	0.4825	0.5088	0.0225	0.0544	0.0769	6.3020×	0.9523
2	500	3.8055	3.8745	0.0876	0.2258	0.3134	12.1438×	0.9822
3	1000	36.2676	36.3036	0.4653	1.2461	1.7115	21.1911×	0.9990
4	1500	138.5206	138.4096	1.7331	3.8700	5.6031	24.7222×	1.0008
5	2000	337.3394	337.3223	4.3605	7.8746	12.2351	27.5714×	1.0001

$l = 50$ with check failure probability $Prob_f \leq \frac{1}{2^{50}}$: **Tradeoff between Cheating Resistance and Client Speedup**

Benchmark		Original MIC	Encrypted MIC_K				Client Speedup	Cloud Efficiency
No.	dimension n	t_{original}	t_{cloud}	t_{client1}	t_{client2}	t_{client}	$t_{\text{original}}/t_{\text{client}}$	$t_{\text{original}}/t_{\text{cloud}}$
1	250	0.4828	0.4916	0.0116	0.1117	0.1233	3.9149×	0.9821
2	500	3.8374	3.8881	0.0648	0.4661	0.5309	7.2280×	0.9870
3	1000	36.2769	36.4662	0.4588	2.4061	2.8649	12.6626×	0.9948
4	1500	138.0207	137.8952	1.7620	6.9443	8.7063	15.8530×	1.0009
5	2000	337.3203	337.6601	4.4533	12.8006	17.2539	19.5504×	0.9990

$l = 80$ with check failure probability $Prob_f \leq \frac{1}{2^{80}}$: **High Cheating Resistance and Low Client Speedup**

Benchmark		Original MIC	Encrypted MIC_K				Client Speedup	Cloud Efficiency
No.	dimension n	t_{original}	t_{cloud}	t_{client1}	t_{client2}	t_{client}	$t_{\text{original}}/t_{\text{client}}$	$t_{\text{original}}/t_{\text{cloud}}$
1	250	0.4971	0.5054	0.0239	0.1707	0.1947	2.5536×	0.9835
2	500	3.7922	3.8659	0.0636	0.7022	0.7658	4.9521×	0.9809
3	1000	37.6828	38.2111	0.4578	3.5890	4.0468	9.3117×	0.9862
4	1500	140.7814	138.9219	1.7289	10.1187	11.8476	11.8827×	1.0134
5	2000	336.4960	338.0702	4.3743	17.7906	22.1649	15.1815×	0.9953

should not increase the time to solve the original MIC. It is desirable that the cloud efficiency is close to 1.

The implementation is done using Matlab 2012a on a workstation equipped with Intel(R) Core(TM) 3.20-GHz CPU and 4-GB RAM. In our experiments, the Guass-Jordan elimination, the most commonly used schoolbook matrix inversion algorithm, is employed by the cloud. All of matrix instances in experiments are generated to be invertible with each entry randomly located in $(0, 1)$. We set $\mathcal{K}_\alpha = \mathcal{K}_\beta = \{1, \ldots, n\}$ and conduct three groups of experiments with $l = 20$, $l = 50$, and $l = 80$.

The choice of parameter l is intuitive. From the proof of Theorem 4, we have that the probability of a false negative is upper bounded by $\frac{1}{2^l}$. Keeping $Prob_f \leq \frac{1}{2^l}$ in mind, if $l = 20$, then the probability of a false negative $Prob_f \leq \frac{1}{2^{20}} \approx 10^{-6}$, this means that a false negative occurs with probability no more than 10^{-6}. In other words, there is one occurrence of a false negative in more than 10^6 trials on average. This probability is a little bit large and a false negative may occur in applications, we label this case as “efficiency priority case.” Likewise, if $l = 80$, a false negative occurs with probability no more than $\frac{1}{2^{80}} \approx 10^{-24}$. This probability is very small and a false negative “nearly cannot” occur in applications, we label this case as “cheating resistance priority case.”

The main performance is shown in Table 5. It can be observed that client speedup is monotonically increasing with matrix dimension n. Outsourcing MIC is able to gain more than 10 times client speedup if n is sufficiently large. It can also be found that $t_{\text{client2}} > t_{\text{client1}}$, which indicates that handling result decryption and verification is more costly. Besides, the cloud efficiency stays close to 1, which is very

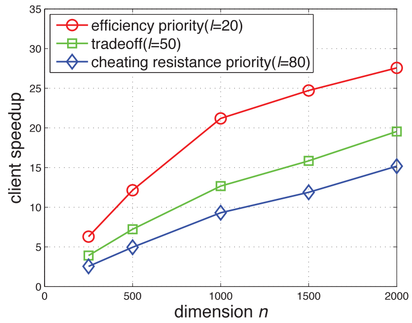


Fig. 2. Comparison of client speedup.

satisfactory. The comparison of client speedup as a function of matrix dimension n is depicted in Fig. 2. It is shown that client speedup in the case of efficiency priority($l = 20$) is much larger than that in the case of cheating resistance priority($l = 80$), which is expected. Note that, in our experiments $n \leq 2,000$, a matrix with dimension $n = 2,000$ is not an unreasonably large matrix. Many real-world applications, for example, MIC from processing 3D graphics or MIC in image watermarking, could easily lead to a large matrix with considerably more than 2,000 dimensions.

Remarkably, the experimental performance really depends on matrix dimension, code compile platform, and the selected algorithm for MIC in the cloud side. If the cloud exploits other faster matrix inversion algorithms as displayed in Table 2, then client speedup will decrease to some extent. However, as long as n goes sufficiently large, the substantial computation savings can always be anticipated by the client due to the clear existence of a gap between $O(n^2)$ and $O(n^\rho)$.

6 Related Work

Secure outsourcing, since its proposal, has stimulated considerable research efforts both from theoretical cryptographers and security engineers. With the advent of cloud and mobile computing age, the theoretical cryptographers' interest in secure outsourcing is persistently increasing, especially after Gentry's first FHE scheme [18] by using an ideal lattice. They often focus on designing a generic protocol that covers all problems, for example, [19], [20]. The generic protocol always involves in applying an FHE scheme, which is a cryptographic primitive that seems to be far from practical. Hence, the generic protocol is currently quite complicated and inefficient. As to security engineers, they often identify some specific problems and design different techniques to mask the original problem to protect input/output privacy. Their protocols always lack formal security treatment and do not handle the important case of result verification, but these protocols are always quite efficient and can be deployed immediately.

Works for Specific Applications. Over the past few decades, many of protocols have been designed for secure outsourcing of some specific applications. The common drawback of the early protocols on secure outsourcing is that they often lack detailed efficiency analysis and evaluation. Also, they do not handle the important case of result verification. Until recently, two secure matrix multiplication outsourcing protocols were introduced in [21] and [22]. The former is built upon the assumptions of two noncolluding servers, making it vulnerable to colluding attacks, while the latter achieves provable security using Shamir's secret sharing [23] technique. But this theoretically elegant protocol still suffers from large amount of communication overhead. After Gentry's breakthrough work on FHE scheme, the research direction is currently shifting to design secure outsourcing protocol in the malicious cloud model rather than in the fully trusted cloud model. Hence, handling result verification becomes a must. Following this trend, several protocols that can handle result verification are proposed, among which there are the secure outsourcing of linear programming [24] and the secure outsourcing of linear equations [25], and so on. Our system model and framework are inherited from these works. Besides, the main idea of this work to protect input privacy is similar to that of [24], i.e., modifying the input matrix so that it looks rather different from the original input.

Functionally Related Work. We would like to give an overview of three kinds of existing work that are conceptually and functionally related to secure outsourcing. The first is secure multiparity computation (SMC), initially introduced by Yao [26] and later developed by Goldwasser et al. [27] and many subsequent researchers. The same to secure outsourcing, SMC can be applied to privacy-preserving cooperative computations. But a major difference with secure outsourcing lies in that SMC does not consider the asymmetry between the resources possessed by cloud and client. This indicates that SMC cannot be applied in secure outsourcing directly. Second is about delegating computation and cheating detection (see [28] for some recent general results). It is shown in this paper that detecting the malicious behaviors of the cloud is often troublesome and costly. The traditional work on cheating detection allows the server to access the original data. However, for data privacy purpose, access to original data is not granted to the cloud server in our secure outsourcing paradigm. Therefore, further research should focus more on the improvements of those previous works in the protection of input/output data privacy. The third is work on server-aided computations, such as [29], [30], [31], to just list a few. One limitation of these protocols is that they are mainly concerned with outsourcing of cryptographic computations like signature and modular exponentiation. The other limitation is that these protocols do not handle result verification.

7 Conclusions

In this paper, we have designed a protocol for outsourcing of MIC to a malicious cloud. We have shown that the proposed protocol simultaneously fulfills the goals of correctness, security (input/output privacy), robust cheating resistance, and high efficiency. With MIC already well rooted in scientific and engineering fields, the proposed protocol can be deployed individually or serve as a primitive building block, based on which some higher level secure outsourcing protocols are constructed. We also introduced a Monte Carlo verification algorithm to handle result verification. Its superiority in designing inexpensive result verification algorithm for secure outsourcing is well demonstrated. Directions to launch further research include: 1) establishing formal security framework for MIC outsourcing problem; 2) adding result verification for some early protocols, which do not handle result verification, as a counteroffensive to malicious cloud; and 3) identifying new meaningful scientific and engineering computational tasks and then designing protocols to solve them.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61170249), the Natural Science Foundation project of CQCSTC (2009BA2024), in part by the State Key Laboratory of Power Transmission Equipment & System Security and New Technology, Chongqing University (2007DA10512711206), in part by the program for Changjiang scholars, and supported by Specialized Research Fund for priority areas for the Doctoral Program of Higher Education. This work was also supported by NPRP Grant 4-1162-1-181 from the Qatar National Research Fund (a member of the Qatar Foundation).

References

[1] G. Brunette and R. Mogull, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1," Cloud Security Alliance, pp. 1-76, 2009.

[2] G.A. Seber and A.J. Lee, *Linear Regression Analysis.* John Wiley & Sons 2012.

[3] S.F. Gibson and B. Mirtich, "A Survey of Deformable Modeling in Computer Graphics," Technical Report TR-97-19, Mitsubishi Electric Research Laboratory, 1997.

[4] R. Tao, X.-Y. Meng, and Y. Wang, "Image Encryption with Multiorders of Fractional Fourier Transforms," *IEEE Trans. Information Forensics and Security,* vol. 5, no. 4, pp. 734-738, Dec. 2010.

[5] X. Zhang, "Lossy Compression and Iterative Reconstruction for Encrypted Image," *IEEE Trans. Information Forensics and Security,* vol. 6, no. 1, pp. 53-58, Mar. 2011.

[6] S. Lee, C.D. Yoo, and T. Kalker, "Reversible Image Watermarking Based on Integer-to-Integer Wavelet Transform," *IEEE Trans. Information Forensics and Security,* vol. 2, no. 3, pp. 321-330, Sept. 2007.

[7] X. Zhang, Z. Qian, Y. Ren, and G. Feng, "Watermarking with Flexible Self-Recovery Quality Based on Compressive Sensing and Compositive Reconstruction," *IEEE Trans. Information Forensics and Security,* vol. 6, no. 4, pp. 1223-1232, Dec. 2011.

[8] Y. Lindell and B. Pinkas, "Secure Multiparty Computation for Privacy-Preserving Data Mining," *J. Privacy and Confidentiality,* vol. 1, no. 1, article 5, 2009.

[9] R. Durstenfeld, "Algorithm 235: Random Permutation," *Comm. the ACM,* vol. 7, no. 7, p. 420, 1964.

[10] D.E. Knuth, *The Art of Computer Programming.* Addison-Wesley, 2006.

[11] R. Freivalds, "Probabilistic Machines Can Use Less Running Time," *Information Processing,* vol. 77, pp. 839-842, 1977.

[12] R. Motwani and P. Raghavan, *Randomized Algorithms.* Cambridge Univ. Press, 1995.

[13] J. Camenisch, S. Hohenberger, and M. Pedersen, "Batch Verification of Short Signatures," *Proc. 26th Ann. Int'l Conf. Advances in Cryptology (EUROCRYPT '07),* pp. 246-263, 2007.

[14] V. Strassen, "Gaussian Elimination is Not Optimal," *Numerische Math.,* vol. 13, no. 4, pp. 354-356, 1969.

[15] D. Coppersmith and S. Winograd, "Matrix Multiplication via Arithmetic Progressions," *J. Symbolic Computation,* vol. 9, no. 3, pp. 251-280, 1990.

[16] V.V. Williams, "Breaking the Coppersmith-Winograd Barrier," unpublished manuscript, Nov. 2011.

[17] C.H. Papadimitriou, *Computational Complexity.* John Wiley & Sons, 2003.

[18] C. Gentry, "A Fully Homomorphic Encryption Scheme," PhD dissertation, Stanford Univ., 2009.

[19] R. Gennaro, C. Gentry, and B. Parno, "Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers," *Proc. 30th Ann. Conf. Advances in Cryptology (CRYPTO '10),* pp. 465-482, 2010.

[20] K. Chung, Y. Kalai, and S. Vadhan, "Improved Delegation of Computation Using Fully Homomorphic Encryption," *Proc. 30th Ann. Conf. Advances in Cryptology (CRYPTO '10),* pp. 483-501, 2010.

[21] D. Benjamin and M.J. Atallah, "Private and Cheating-Free Outsourcing of Algebraic Computations," *Proc. Sixth Ann. Conf. Privacy, Security and Trust (PST '08),* pp. 240-245, 2008.

[22] M. Atallah and K. Frikken, "Securely Outsourcing Linear Algebra Computations," *Proc. Fifth ACM Symp. Information, Computer and Comm. Security,* pp. 48-59, 2010.

[23] A. Shamir, "How to Share a Secret," *Comm. the ACM,* vol. 22, no. 11, pp. 612-613, 1979.

[24] C. Wang, K. Ren, and J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," *Proc. IEEE INFOCOM,* 2011.

[25] C. Wang, Q. Wang, K. Ren, and J. Wang, "Harnessing the Cloud for Securely Outsourcing Large-Scale Systems of Linear Equations," *IEEE Trans. Parallel and Distributed Systems,* vol. 24, no. 6, pp. 1172-1181, June 2013.

[26] A. Yao, "Protocols for Secure Computations," *Proc. 23rd Ann. Symp. Foundations of Computer Science,* pp. 160-164, 1982.

[27] S. Goldwasser, S. Micali, and A. Wigderson, "How to Play Any Mental Game, or a Completeness Theorem for Protocols with an Honest Majority," *Proc. 19th Ann. ACM Symp. Theory of Computing,* vol. 87, pp. 218-229, 1987.

[28] S. Goldwasser, Y. Kalai, and G. Rothblum, "Delegating Computation: Interactive Proofs for Muggles," *Proc. 40th Ann. ACM Symp. Theory of Computing,* pp. 113-122, 2008.

[29] T. Matsumoto, K. Kato, and H. Imai, "Speeding Up Secret Computations with Insecure Auxiliary Devices," *Proc. Advances in Cryptology (CRYPTO '88),* pp. 497-506, 1990.

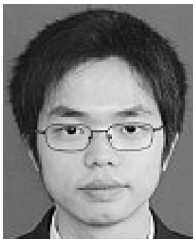
[30] S. Hohenberger and A. Lysyanskaya, "How to Securely Outsource Cryptographic Computations," *Proc. Second Int'l Conf. Theory of Cryptography,* pp. 264-282, 2005.

[31] S. Kawamura and A. Shimbo, "Fast Server-Aided Secret Computation Protocols for Modular Exponentiation," *IEEE J. Selected Areas in Comm.,* vol. 11, no. 5, pp. 778-784, June 1993.

Xinyu Lei received the BS degree in computing science from Chongqing University, China, in 2010. He is working toward the PhD degree in computer science at Chongqing University, and is a visiting scholar at Texas A&M University at Qatar, Doha. His research interests include information security and algorithms.

Xiaofeng Liao received the BS and MS degrees in mathematics from Sichuan University, Chengdu, China, in 1986 and 1992, respectively, and the PhD degree in circuits and systems from the University of Electronic Science and Technology of China in 1997. His current research interests include neural networks, nonlinear dynamical systems, bifurcation and chaos, and cryptography. He is a senior member of the IEEE.

Tingwen Huang received the BS degree from Southwest Normal University (now Southwest University), China, in 1990, the MS degree from Sichuan University, China, in 1993, and the PhD degree from Texas A&M University, College Station, in 2002. He was the president of Asia Pacific Neural Networks Assembly in 2012. He is currently an associate editor for the *IEEE Transactions on Neural Networks and Learning Systems.*



Huaqing Li received the BS degree from Chongqing University of Posts and Telecommunications in Information and Computation Science, China, in 2009. He is currently working toward the PhD degree in computer science at Chongqing University. His research interests include nonlinear dynamical systems, bifurcation and chaos, and consensus of multiagent systems.

Chunqiang Hu received the BS degree in computer science from Southwest University, Chongqing, China, in 2006, and the MS degree in computer science from Chongqing University, China, in 2009. He is working toward the PhD degree in computer science at Chongqing University and is a visiting scholar at the George Washington University. His research interests include wireless and mobile security, secret sharing, and cryptography.