SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2018, Vol. 61 032112:1–032112:9
https://doi.org/10.1007/s11432-017-9208-3

Privacy-preserving large-scale systems of linear equations in outsourcing storage and computation

Dongmei LI[1], Xiaolei DONG[2*], Zhenfu CAO[2] & Haijiang WANG[1]

[1]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai* 200062*, China*

Received 18 March 2017/Revised 18 July 2017/Accepted 16 August 2017/Published online 2 February 2018

Abstract Along with the prevalence of cloud computing, it can be realised to efficiently outsource costly storage or computations to cloud servers. Recently, secure outsourcing mechanism has received more and more attention. We focus on secure outsourcing storage and computation for large-scale systems of linear equations (LEs) in this paper. Firstly, we construct a new efficient matrix encryption scheme. Then we exploit this encryption scheme to develop a new algorithm which can implement outsourcing storage and computation for large-scale linear equations in the semi-honest setting. Compared with the previous work, the proposed algorithm requires lower storage overhead and is with competitive efficiency.

Keywords cloud computing, privacy-preserving, linear equations, encryption, security

Citation Li D M, Dong X L, Cao Z F, et al. Privacy-preserving large-scale systems of linear equations in outsourcing storage and computation. Sci China Inf Sci, 2018, 61(3): 032112, https://doi.org/10.1007/s11432-017-9208-3

1 Introduction

Along with the prevalence of cloud technology, it becomes possible for clients with weaker storing or computing power to store and process data [1,2]. Therefore, the enterprises and individuals are beneficial to lessen capital outlays in hardware and software overheads. Secure outsourcing storage and computation has become more and more attractive [3, 4]. However, it is of great importance to ensure data security and privacy [5–8].

The large-scale system of linear equations (LEs) $Ax = b$ is a basic algebraic problem in science and engineering areas (such as finite element analysis, optimal design and image recognition), which often consumes a lot of storage and computation [9]. There are many application problems that will result in very large-scale systems of LEs reaching up to hundreds of thousands or more [10]. In nowadays, some secure outsourcing solutions for large-scale systems of LEs have significantly developed [11–13]. We present a new privacy-preserving framework for large-scale systems of LEs in outsourcing storage and computation showed in Figure 1.

1.1 Related work

As more and more users are ready to outsource their encrypted data to cloud servers for storage and computation, it is necessary to ensure data manipulability with privacy-preserving. In recent years, there are many studies on securely outsourcing storage and computation. A general mechanism of securely outsourcing computation has proposed by Gennaro et al. [14], which is implemented by garbled circuits [15]

* Corresponding author (email: dongxiaolei@sei.ecnu.edu.cn)

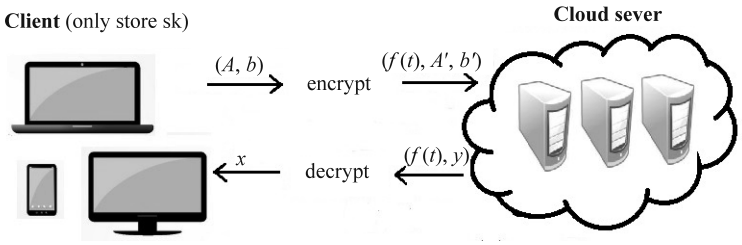

Figure 1 Framework for our scheme.

and fully homomorphic encryption scheme [16]. Yet, it is not practical as a result of high complexity. A list of customized solutions are proposed for securely outsourcing specific computations [17–19].

Some securely outsourcing schemes have been presented to solve LEs. Lei et al. [20] have proposed a secure matrix inversion algorithm by using matrix permutations. Wang et al. [11] have presented an iterative algorithm for solving LEs by using Paillier encryption cryptosystem [21]. Due to use homomorphic encryption, it performs computations with higher computational complexity. Besides, it requires multiple rounds interactions and the privacy will be revealed when the number of iterations reaches or exceeds n. In addition, Chen et al. [13] have put forward a new secure outsourcing algorithm for LEs, which utilizes the sparse matrix and only requires one round interactions. However, it requires higher storage overhead.

1.2 Contributions

Our proposed algorithm works under semi-honest setting where the cloud server is honest but curious. Compared with the previous work, our solution has the following contributions.

(1) With respect to matrix operation, we present a new efficient matrix encryption scheme, and then we prove its security under chosen-plaintext adversaries (CPA) existing.

(2) In our scheme, the client can only store his secret key but does not need to store LEs. We realize the outsourcing storage and computation for large-scale systems of LEs.

(3) Our solution reduces the client's communication overhead, which only requires one round communication between the client and cloud server.

1.3 Organization

The rest of the paper is organized as follows. Section 2 introduces our efficient matrix encryption scheme and gives security proof. Then we draw into the definition of securely outsourcing computation scheme in Section 3. Section 4 discusses the practical implementation issues in details, followed by Section 5 which gives security analysis. Section 6 makes the performance evaluation. Finally, Section 7 gives out the conclusion.

2 Efficient matrix encryption

In this section, a new model of matrix encryption is described. In matrix encryption scheme, the plaintext and the ciphertext are matrices. As in [22], the security of our encryption scheme relies on trapdoor permutation. A trapdoor permutation generator is a probabilistic polynomial time (PPT) algorithm, which inputs 1^k and outputs $(f; f^{-1}; d)$ [23]. Here f is a permutation function and f^{-1} is the inverse. There are some good examples such as RSA [24], squaring modulo [25, 26], or variations of it [23].

2.1 Encryption scheme

Now we introduce an efficient matrix encryption scheme Π_M, which is composed of three algorithms **(KeyGen, Enc, Dec)**.

KeyGen(k). This algorithm selects a trapdoor permutation generator to generate $(f; f^{-1})$ and chooses a hash function $G : \{0,1\}^{kn} \to \mathbb{R}^{n\times n}$. Then it outputs the public key $\mathrm{pk} = (f, G)$ and the corresponding secret key $\mathrm{sk} = f^{-1}$.

Enc(pk, X). Suppose $X \in \mathbb{R}^{m\times m}$ $(m < n)$ is the matrix to be encrypted. This algorithm randomly pads the matrix X before encrypting. It chooses three random matrices $R_1 \in \mathbb{R}^{(n-m)\times(n-m)}$, $R_2 \in \mathbb{R}^{(n-m)\times m}$, $R_3 \in \mathbb{R}^{m\times(n-m)}$ and interprets $X' = \begin{bmatrix} R_1 & R_2 \\ R_3 & X \end{bmatrix}$ as a matrix of $\mathbb{R}^{n\times n}$. Then this algorithm selects a random vector $t = \langle t_1, t_2, \ldots, t_n\rangle \leftarrow \{0,1\}^{kn}$ and sets $T = G(t)$, where $T \in \mathbb{R}^{n\times n}$ is a matrix. If T is non-invertible, random vector t is re-selected until T is invertible. Then it computes $C_1 = f(t) = \langle f(t_1), f(t_2), \ldots, f(t_n)\rangle$, $C_2 = TX'$. The cipher is $C = (C_1, C_2)$.

Dec(sk, C). Given a cipher $C = (C_1, C_2)$, this algorithm computes $t = \langle t_1, t_2, \ldots, t_n\rangle = f^{-1}(C_1)$ and sets $T = G(t)$. It computes $X' = T^{-1}C_2$ and outputs the lower right $m \times m$ matrix of X' as the decrypted matrix.

A matrix encryption scheme should be both correct and secure. A scheme is correct if the output matrix of the decryption algorithm can recover the input matrix of the encryption algorithm. More formally:

Definition 1 (Correctness)**.** A matrix encryption scheme is correct if there exists a negligible function ϵ such that for every k, every $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(k)$, and every matrix X in the appropriate underlying plain space, it holds that $\Pr[\mathrm{Dec}(\mathrm{sk}, \mathrm{Enc}(\mathrm{pk}, X)) \neq X] \leqslant \epsilon(k)$.

Easily, we can prove that the above scheme is an encryption scheme. That is to say, the decryption of an encryption of a matrix recovers the matrix. We have $X' = \begin{bmatrix} R_1 & R_2 \\ R_3 & X \end{bmatrix}$, $f^{-1}(C_1) = f^{-1}(f(t)) = t$, $T = G(t)$, $T^{-1}C_2 = T^{-1}TX' = X'$. It is easy to recover X from X'.

2.2 Construction of hash function G

The hash function G can be constructed by a general hash function. Next we give an example of construction. Suppose H is a fixed-length collision-resistant hash function that takes an input of $2l(k)$ length and an output of $l(k)$ length, and B is a function, the output length of which is lower than $l(k)$ when taking the same length of $2l(k)$ as input. We can construct a hash function G for input of a n-dimension vector $\langle v_1, v_2, \ldots, v_n\rangle$ and output of a $n \times n$ matrix T as Algorithm 1.

Algorithm 1

Input $\langle v_1, v_2, \ldots, v_n\rangle$.
for $i = 1, 2, \ldots, n$
  $t_{i,1} = H(v_i)$;
  for $j = 2, 3, \ldots, n$
    $t_{i,j} = H(B(v_i)||t_{i,j-1})$;
  endfor
endfor
for $i = 1, 2, \ldots, n$
  for $j = 1, 2, \ldots, n$
    $T_{ij} = t_{i,j}$;
  endfor
endfor
Output T.

2.3 Proof of security

Our construction depends on trapdoor permutation. Now, we introduce some basic definitions and give security proof of the above encryption scheme.

Definition 2 (Trapdoor permutation generator [23])**.** Given a security parameter k, a trapdoor permutation generator is defined as a PPT algorithm $\mathcal{G}_p$ which inputs 1^k and outputs $(f; f^{-1})$. For any PPT adversary $\mathcal{A}_p$, there exists a negligible function ϵ such that

$$\Pr[(f, f^{-1}) \leftarrow \mathcal{G}_p(1^k); x \leftarrow \{0,1\}^k; y \leftarrow f(x) : \mathcal{A}_p(f, y) = x] \leqslant \epsilon(k).$$

Intuitively, a matrix encryption scheme is CPA-secure if a CPA adversary cannot persuade the decryption algorithm to accept an incorrect output. Specifically, consider the following experiment defined for matrix encryption π and PPT adversary $\mathcal{A}$.

Experiment $\mathrm{PubK}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(k)$:

(1) Run KeyGen(k) to obtain (pk, sk).

(2) Send pk to the adversary $\mathcal{A}$ and assign oracle access of $\mathrm{Enc}_{\mathrm{pk}}(\cdot)$ to $\mathcal{A}$. It outputs a pair of matrix (X_0, X_1), where X_0 and X_1 have the same size.

(3) Select a random bit $b \leftarrow \{0,1\}$. Compute $C \leftarrow \mathrm{Enc}_{\mathrm{pk}}(x_b)$ and give it to $\mathcal{A}$.

(4) $\mathcal{A}$ continues to have oracle access to $\mathrm{Enc}_{\mathrm{pk}}(\cdot)$ and outputs a bit b'.

(5) Output 1 if $b' = b$, and 0 otherwise.

Definition 3 (Security)**.** A matrix encryption scheme π is CPA-secure under the random oracle model, if for any PPT chosen-plaintext adversary $\mathcal{A}$ there exists a negligible function ϵ such that

$$\Pr[\mathrm{PubK}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(k) = 1] \leqslant \frac{1}{2} + \epsilon(k).$$

Note that the probability of distinguishing the encryptions between X_0 and X_1 is negligible.

Theorem 1. The proposed efficient matrix encryption scheme is CPA-secure in the random oracle model.

Proof. We prove this theorem by contradiction.

Suppose $\mathcal{A}_{\mathrm{CPA}} = (F, A)$ is a PPT chosen-plaintext adversary that defeats the protocol with nonnegligible advantage $\lambda(k)$, which is $\Pr[\mathrm{PubK}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(k) = 1] = \frac{1}{2} + \lambda(k)$. F's job is to come up with a pair of matrices X_0 and X_1 (with the same size) such that when a ciphertext α is sent to A, A cannot guess well which matrix the decryption of α is, where α is supposed to be the encryption of X_0 or X_1. The random oracle G is defined in the encryption scheme.

Given $(f, f^{-1}) \leftarrow \mathcal{G}_p(1^k)$, $r \leftarrow \{0,1\}^{kn}$, and $y \leftarrow f(r)$, we can construct an algorithm $\mathcal{A}_p(f, y)$ to compute $f^{-1}(y)$ using $\mathcal{A}_{\mathrm{CPA}}$:

It simulates the oracle G and samples $(X_0, X_1) \leftarrow F^G(\mathrm{Enc})$.

(1) If G is once queried by a vector $t = \langle t_1, t_2, \ldots, t_n \rangle$ which satisfies $f(t_i) = y$, $\mathcal{A}_p$ outputs t_i as the result.

(2) Otherwise, it chooses $S \leftarrow \mathbb{R}^{n \times n}$, $Y \leftarrow \mathbb{R}^{n-1}$ and sets $(C_1, C_2) = ((y, Y), S)$. Then $\mathcal{A}_p$ simulates $A(\mathrm{Enc}, X_0, X_1, C_1, C_2)$ until it finds an oracle query $s = \langle s_1, s_2, \ldots, s_n \rangle$ which satisfies $f(s_i) = y$. $\mathcal{A}_p$ outputs s_i. Let A_q be the event that A asks the query $s_i = f^{-1}(y)$. We have

$$\Pr[A \text{ succeeds}] = \Pr[A \text{ succeeds}|A_q] \cdot \Pr[A_q] + \Pr[A \text{ succeeds}|\bar{A}_q] \cdot \Pr[\bar{A}_q],$$

$$\Pr[A \text{ succeeds}] = \frac{1}{2} + \lambda(k),$$

$$\Pr[A \text{ succeeds}|A_q] \cdot \Pr[A_q] + \Pr[A \text{ succeeds}|\bar{A}_q] \cdot \Pr[\bar{A}_q] \leqslant \frac{1}{2} + \Pr[A_q].$$

So $\Pr[A_q] \geqslant \lambda(k)$.

The advantage of $\mathcal{A}_p$ is $\Pr[(f, f^{-1}) \leftarrow \mathcal{G}_p(1^k);\ r \leftarrow \{0,1\}^{kn};\ y \leftarrow f(r) : \mathcal{A}_p(f, d, y) = r] = \Pr[G] + \Pr[A_q] \geqslant \Pr[A_q] \geqslant \lambda(k)$, which is nonnegligible. The deduction is contrast to the definition of trapdoor permutations.

3 Securely outsourcing computation scheme

3.1 Formal definition

The formal definition of secure outsourcing computation was firstly proposed by Gennaro et al. [14]. A secure outsourcing computation scheme is composed of four algorithms **(KeyGen, ProbGen, Compute, Solve)**.

KeyGen$(F, k) \rightarrow (\mathrm{pk}, \mathrm{sk})$. Given the security parameter k and the target function F, this algorithm outputs a public key pk and its corresponding secret key sk.

ProbGen$(\mathrm{sk}, x) \rightarrow (\sigma_x, \tau_x)$. Given a private key sk and a function input x, this algorithm encodes x into a public value σ_x and a secret value τ_x by using sk.

Compute$(\mathrm{pk}, F, \sigma_x) \rightarrow \sigma_y$. Given the client's public key pk and the encoded input σ_x of the function F, this algorithm outputs an encoded value of the function's output σ_y such that $y = F(x)$.

Solve$(\mathrm{sk}, \sigma_y, \tau_x) \rightarrow y$ or $\perp$. Using the secret key sk, this algorithm decodes σ_y into the output of the function $y = F(x)$ or outputs $\perp$ when σ_y is not a valid encoding of $F(x)$.

3.2 Security requirements

Now we lead into some security requirements for outsourcing computation.

Definition 4 (Correctness [14])**.** An outsourcing computation scheme OC is correct if for any choice of function F, the key generation algorithm produces keys $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(F, k)$ which satisfy $\forall x \in \mathrm{Domain}(F)$, $(\sigma_x, \tau_x) \leftarrow \mathrm{ProbGen}(\mathrm{sk}, x)$ and $\sigma_y \leftarrow \mathrm{Compute}(\mathrm{pk}, F, \sigma_x)$, such that $y = F(x) \leftarrow \mathrm{Solve}(\mathrm{sk}, \sigma_y, \tau_x)$.

A outsourcing computation scheme is secure if an adversary cannot forge a pair of input and output.

Definition 5 (Security [14])**.** Let k be a security parameter. If for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible, we say that an outsourcing computation scheme OC is secure for a function F. That is to say, the advantage of an adversary $\mathcal{A}$ in the following experiment satisfies $\mathrm{Adv}_{\mathcal{A}}(\mathrm{OC}, F, k) \leqslant \epsilon(k)$ where $\epsilon(k)$ is a negligible function and $\mathrm{Adv}_{\mathcal{A}}(\mathrm{OC}, F, k) = \Pr[\mathrm{PubK}_{\mathcal{A}}(\mathrm{OC}, F, k) = 1]$.

Experiment $\mathrm{PubK}_{\mathcal{A}}(\mathrm{OC}, F, k)$:

(1) Run $\mathrm{KeyGen}(F, k)$ to obtain $(\mathrm{pk}, \mathrm{sk})$.

(2) Send pk to the adversary $\mathcal{A}$ and assign oracle access of $\mathrm{ProbGen}_{\mathrm{sk}}(\cdot)$ to $\mathcal{A}$. The adversary $\mathcal{A}$ can perform q queries, and each query can obtain $x_i = \mathcal{A}(\mathrm{pk}, x_1, \sigma_1, \ldots, x_{i-1}, \sigma_{i-1})$ and $(\sigma_i, \tau_i) \leftarrow \mathrm{ProbGen}(\mathrm{sk}, x_i)$. After q queries, the adversary $\mathcal{A}$ imitates a pair $(x*, y*)$ where $\sigma_{y*} \neq \sigma_{x_i}$ for $i = 1, \ldots, q$.

(3) Output 1 if $y* \neq \perp$ and $y* = F(x*)$, and 0 otherwise.

It is desirable that the outsouced scheme protects the secrecy of inputs and outputs given to the clients. The definition of input privacy is based on typical theory of indistinguishability. That is, given two inputs x_0, x_1 and a output y, the adversary cannot distinguish between x_0 and x_1 according to output y. Input privacy yields output privacy. An outsourcing computation scheme is private when the public outputs of the problem generation algorithm ProbGen over two different inputs are indistinguishable.

Definition 6 (Privacy [14])**.** Let k be a security parameter. An outsourcing computation scheme OC is said to be private for a function F, if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{\mathcal{A}}(\mathrm{OC}, F, k) = \mid \Pr[\mathrm{PrivK}_{\mathcal{A}}(\mathrm{OC}, F, k) = 1] - \frac{1}{2} \mid \leqslant \epsilon(k)$, where $\epsilon(k)$ is negligible. The oracle $\mathrm{PubProbGen}_{\mathrm{sk}}(x)$ calls $\mathrm{ProbGen}_{\mathrm{sk}}(x)$ to obtain (σ_x, τ_x) and returns only the public part σ_x. The experiment $\mathrm{PrivK}_{\mathcal{A}}(\mathrm{OC}, F, k)$ is defined as follows.

Experiment $\mathrm{PrivK}_{\mathcal{A}}(\mathrm{OC}, F, k)$:

(1) Run $\mathrm{KeyGen}(F, k)$ to obtain $(\mathrm{pk}, \mathrm{sk})$.

(2) Send 1^k to the adversary $\mathcal{A}$ and assign oracle access of $\mathrm{PubProbGen}_{\mathrm{sk}}(\cdot)$ to $\mathcal{A}$. It outputs (x_0, x_1), where x_0 and x_1 has the same length.

(3) Choose a random bit $b \leftarrow \{0, 1\}$, and compute $(\sigma_b, \tau_b) \leftarrow \mathrm{ProbGen}_{\mathrm{sk}}(x_b)$. Then σ_b is given to $\mathcal{A}$.

(4) $\mathcal{A}$ continues to have oracle access to $\mathrm{PubProbGen}_{\mathrm{sk}}(\cdot)$ and outputs a bit b'.

(5) Output 1 if $b' = b$, and 0 otherwise.

Definition 7 (α-Efficiency [27])**.** An outsourcing computation scheme OC is α-efficient for a function F if for all inputs x, the running time of OC is less than or equal to an α-multiplicative factor of the running time of $F(x)$.

4 Secure outsourcing linear equations

Given LEs $Ax = b$, where coefficient matrix $A \in \mathbb{R}^{m\times m}$ and constant terms $b \in \mathbb{R}^m$, the output of LEs is a vector $x \in \mathbb{R}^m$. In this section, we give detailed solutions for outsourcing computation and storage.

4.1 Outsourcing computation

A secure outsourcing linear equations scheme (OLE) consists of a four-tuple **(KeyGen, ProbGen, Compute, Solve)** as follows.

KeyGen(k). Π_M.KeyGen$(k) \to (\mathrm{pk}, \mathrm{sk})$.

ProbGen(sk, A, b). Suppose $n > m$. This algorithm randomly pads A, b firstly. It chooses two random matrices $R_1 \in \mathbb{R}^{(n-m)\times(n-m)}$, $R_2 \in \mathbb{R}^{(n-m)\times m}$, a zero matrix $O \in \mathbb{R}^{m\times(n-m)}$ and a random vector $R_3 \in \mathbb{R}^{n-m}$. It interprets $\hat{A} = \begin{bmatrix} R_1 & R_2 \\ O & A \end{bmatrix}$ as a matrix of $\mathbb{R}^{n\times n}$ and $\hat{b} = \begin{bmatrix} R_3 \\ b \end{bmatrix}$ as a vector of $\mathbb{R}^n$. It selects a random $v = \langle v_1, v_2, \ldots, v_n\rangle \leftarrow (d(1^k))^n$ and generates $T' = G(v)$ where T' is a matrix. If the diagonal of matrix T' contains zero, v is re-selected until the diagonal of matrix T' does not contain zero. For efficient calculations, we choose $T = \mathrm{diag}(T')$ where T is the diagonal matrix of T'. It computes $A' = T\hat{A}T,\ b' = T\hat{b}$.

Compute(pk, A', b'). Using the client's public key $\mathrm{pk} = f$, this algorithm encrypts the vector v as $f(v)$ and outsources $(f(v), A', b')$ to the server. Receiving a triple $(f(v), A', b')$ from the client, the server solves the linear equation $A'y = b'$ yielding y.

Solve(sk, y). Receiving $(f(v), y)$ from the server, this algorithm decrypts and recovers the vector v as $f^{-1}(f(v)) = v$ using the secret key $\mathrm{sk} = f^{-1}$. Then it computes diagonal matrix T. Finally, it computes $x' = Ty$ and converts the solution of the original LEs as the lower m-vector of x'.

Remark. The computation complexity for matrix multiplication is $O(n^3)$, the user cannot carry out expensive computation locally. For the user, the matrix multiplication $T\hat{A}T$ can be efficiently calculated when T is a sparse matrix. The computation complexity is $O(n^2)$. Since the LEs have n unknowns, we choose T as the diagonal matrix, which is enough to guarantee the privacy.

4.2 Outsourcing storage

In our system, a triple $(f(v), A', b')$ is outsourced to the server. Thus the client can only restore the secret key which is able to recover the solution of LEs. That is to say, the client can outsource to store the LEs $Ax = b$ to the server as the triple $(f(v), A', b')$, which implements outsourcing storage of LEs.

5 Security analysis

A secure outsourcing scheme should be satisfied to four security requirements motioned in Section 3. Now we prove that the proposed scheme is a secure outsourcing scheme. That is to say, it meets with security requirements for outsourcing computation.

Theorem 2 (Correctness)**.** Our outsourcing computation scheme OLE is correct for large-scale systems of LEs.

Proof. In our outsourcing scheme OLE, the server solves LEs $A'y = b'$, where $A' = T\hat{A}T$, $b' = T\hat{b}$. Thus $T\hat{A}Ty = T\hat{b} \Rightarrow \hat{A}Ty = \hat{b} \Rightarrow \begin{bmatrix} R_1 & R_2 \\ O & A \end{bmatrix} T \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} R_3 \\ b \end{bmatrix} \Rightarrow ATy_2 = b \Rightarrow Ty_2 = x$, where $y_1 \in \mathbb{R}^{n-m},\ y_2 \in \mathbb{R}^m$. Our outsourcing computation scheme OLE correctly solves large-scale systems of LEs $Ax = b$.

Theorem 3 (Security)**.** Our outsourcing computation scheme OLE is CPA-secure for large-scale systems of LEs in the random oracle model.

Proof. We prove this theorem by reduction to absurdity.

Suppose $\mathcal{A}_{\mathrm{CPA}} = (F, A)$ is a PPT chosen-plaintext adversary that defeats our scheme OLE with non-negligible advantage $\lambda(k)$, which is $\Pr[v \leftarrow \mathbb{R}^n; (\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KeyGen}(k); (A_0, A_1) \leftarrow F(\mathrm{ProbGen}); (b_0, b_1) \leftarrow \mathbb{R}^m; \sigma \leftarrow \{0, 1\}; (f(v), A', b') \leftarrow \mathrm{ProbGen}(\mathrm{sk}, A_\sigma, b_\sigma) : A(\mathrm{ProbGen}, A_0, A_1, b_0, b_1, f(v), A', b') = \sigma] = \frac{1}{2} + \lambda(k)$. F's job is to come up with a pair of matrices A_0 and A_1 (with the same size) such that when a

ciphertext α is sent to A, A cannot guess well which matrix the decryption of α is, where α is supposed to be the encryption of A_0 or A_1. The random oracle G is defined in the encryption scheme.

Let $\mathcal{G}_p$ be a trapdoor permutation generator $(f, f^{-1}, d) \leftarrow \mathcal{G}_p(1^k)$; $r \leftarrow d(1^k)$; $y \leftarrow f(r)$. We can construct an algorithm $\mathcal{A}_p(f, d, y)$ to compute $f^{-1}(y)$ by $\mathcal{A}_{\mathrm{CPA}} = (F, A)$ as follows.

It simulates the oracle G and samples $(A_0, A_1) \leftarrow F(\mathrm{ProbGen})$.

(1) If G is once queried by a vector $v = \langle v_1, v_2, \ldots, v_n \rangle$ such that $f(v_i) = y$, then $\mathcal{A}_p$ outputs v_i as the result.

(2) Otherwise, it chooses $S \leftarrow \mathbb{R}^{n\times n}$, $Y \leftarrow \mathbb{R}^{n-1}$ and sets $f(v) = (y, Y)$. Then $\mathcal{A}_p$ simulates $A(\mathrm{ProbGen}, A_0, A_1, b_0, b_1, f(v), A', b')$ until it finds an oracle query $v = \langle v_1, v_2, \ldots, v_n \rangle$ which satisfies $f(v_i) = y$. $\mathcal{A}_p$ outputs v_i. Let A_q be the event that A asks the query $v_i = f^{-1}(y)$. We have

$$\Pr[A \text{ succeeds}] = \Pr[A \text{ succeeds}|A_q] \cdot \Pr[A_q] + \Pr[A \text{ succeeds}|\bar{A}_q] \cdot \Pr[\bar{A}_q],$$

$$\Pr[A \text{ succeeds}] = \frac{1}{2} + \lambda(k),$$

$$\Pr[A \text{ succeeds}|A_q] \cdot \Pr[A_q] + \Pr[A \text{ succeeds}|\bar{A}_q] \cdot \Pr[\bar{A}_q] \leqslant \frac{1}{2} + \Pr[A_q].$$

So $\Pr[A_q] \geqslant \lambda(k)$.

The advantage of $\mathcal{A}_p$ is $\Pr[(f, f^{-1}, d) \leftarrow \mathcal{G}_p(1^k); r \leftarrow d(1^k); y \leftarrow f(r) : \mathcal{A}_p(f, d, y) = r] = \Pr[G] + \Pr[A_q] \geqslant \Pr[A_q] \geqslant \lambda(k)$, which is non-negligible. The deduction is contrast to the definition of trapdoor permutations.

Theorem 4 (Privacy)**.** Our outsourcing computation scheme OLE is private for input A, b and output x.

Proof. Suppose $\mathcal{A}$ is a PPT adversary that defeats our scheme OLE. It can only know $f(v)$, A', b' and y throughout the whole implementation. Since f is a trapdoor permutation, the advantage of $\mathcal{A}$ knows v is negligible. Then the advantage of $\mathcal{A}$ knows T is negligible. Let $T' \xleftarrow{R} \mathbb{R}^{n\times n}$. The advantage of $\mathcal{A}$ to distinguish between T and T' is negligible. Besides, we have $A' = T\hat{A}T$, $b' = T\hat{b}$, and $x' = Ty$, thus $\hat{A}$, $\hat{b}$, x' are private to $\mathcal{A}$. It can be deduced that A, b, x are private to $\mathcal{A}$. The proposed scheme protects the input privacy and output privacy quite well.

Theorem 5 (Efficiency)**.** If the computational complexity of trapdoor permutation f is not more than $O(n^2)$, our outsourcing computation scheme OLE is $\frac{1}{n}$-efficient for large-scale systems of LEs $Ax = b$.

Proof. In our outsourcing computation scheme OLE, the client needs to perform $T\hat{A}T$ and $T\hat{b}$, which takes $O(n^2)$ computations in the condition that we choose T as a diagonal matrix. On the other hand, it takes $O(n^3)$ computations to solve the LEs directly. Thus, our OLE is an $\frac{1}{n}$-efficient implementation for large-scale systems of LEs when the computational complexity of trapdoor permutation f is not more than $O(n^2)$.

6 Evaluation

In our scheme, the user only needs to encrypt-then-outsource his data to the cloud server. The server is free to choose the method of solving LEs. So we only evaluate the performance of the user.

6.1 Comprehensive comparison

Considering that one regular exponentiation operation with an exponent of length $\|N\|$ requires $1.5\|N\|$ multiplications (e.g., if the length of r is $\|N\|$, it requires $1.5\|N\|$ multiplications to compute g^r) [28]. Let k be the security parameter. For the Paillier cryptosystem, Enc needs $1.5k \times 1.5k = 2.25k^2$ multiplications to encrypt a message, and Dec also needs $2.25k^2$ multiplications to encrypt a message. For the RSA cryptosystem, Enc needs $1.5k$ multiplications to encrypt a message, Dec also needs $1.5k$ multiplications to encrypt a message. Suppose we choose RSA as the trapdoor permutation. We compare our scheme with the existing schemes [11–13]. The proposed scheme is competitive in efficiency. Moreover, we need less storage cost compared with previous schemes. Let M be an operation of multiplication, L be the

Table 1 Comparison with the existing schemes [11–13]

	Scheme [11]	Scheme [12]	Scheme [13]	Our scheme
Storage cost	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$
Computation of ProbGen	$(2n^2+n)M+2.25k^2n^2M$	$(4n^2+4n)M$	$((2\lambda+1)n^2+\lambda n)M$	$(2n^2+n)M+1.5knM$
Computation of solve	$2.25k^2LnM$	$(22Ln-4L)M$	λnM	$1.5knM+nM$
Communication round	L	L	1	1

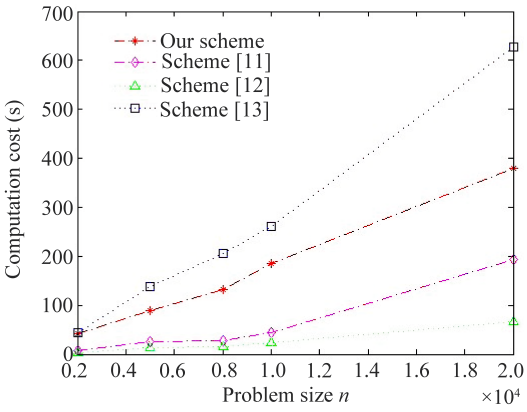

Figure 2 (Color online) Computation cost in the user side.

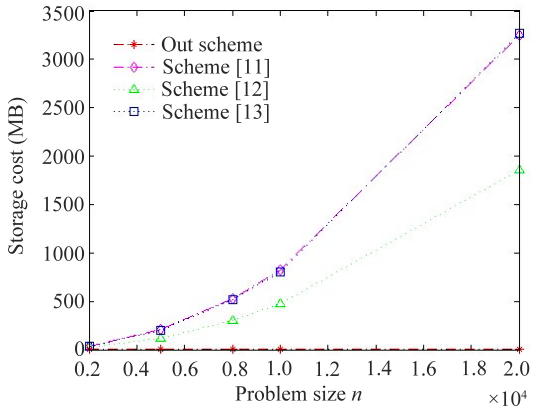

Figure 3 (Color online) Storage cost in the user side.

maximum number of iterations in [11,12], and λ be the maximum number of non-zero elements for each row (or column) of the chosen sparse matrices in [13]. The detail comparison can be seen in Table 1.

6.2 Experiment results

Theoretical analysis shows that our proposed scheme benefits the user. We further carry out experiments to verify the practical efficiency of our work. We perform the experiments on a personal computer (PC) with 3.6 GHz processor and 8 GB RAM memory. We choose 1024-bit RSA cryptosystem as the trapdoor permutation. In the experiments, we choose SHA-256 instead of the fixed-length hash function H. The user's computation cost is compared with previous schemes (Figure 2). We also evaluate the storage cost (Figure 3). The experiments show that schemes in [11,12] are more efficient, but they need frequent interaction due to iterative computation. No real time interaction is required in our scheme and scheme [13]. Moreover, our scheme attains minimum storage overhead.

7 Conclusion

In this paper, we present a framework for privacy-preserving large-scale systems of LEs in outsourcing storage and computation. It has been also proved that the construction is CPA-secure under the semi-honest adversaries. The user can flexibly outsource his data and computation to the cloud and do not need real-time online. Compared with previous work, our solution reduces storage cost with competitive computational overhead.

Privacy-protected outsourcing storage and computation still has challenging problems and requires more investigation. We believe that the presented work should be an important step toward privacy-preserving applications in an environment where privacy is a major concern. New solutions will emerge to address the problems of security in cloud storage and computation.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. 61371083, 61373154, 61632012, 61672239), Prioritized Development Projects through the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20130073130004), and Shanghai

High-Tech Field Project (Grant No. 16511101400).

References

1 Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing. 2009. http://www.cloudsecurityalliance.org

2 Wang C, Ren K, Wang J. Secure optimization computation outsourcing in cloud computing: a case study of linear programming. IEEE Comput Soc, 2016, 65: 216–229

3 Loftus J, Smart N P. Secure outsourced computation. In: Proceedings of International Conference on Cryptology in Africa, Dakar, 2011. 1–20

4 Zhou J, Cao Z F, Dong X L, et al. EVOC: more efficient verifiable outsourced computation from any one-way trapdoor function. In: Proceedings of IEEE International Conference on Communications, London, 2015. 7444–7449

5 Pearson S, Benameur A. Privacy, security and trust issues arising from cloud computing. In: Proceeding of IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, 2010. 693–702

6 Zissis D, Lekkas D. Addressing cloud computing security issues. Future Gener Comput Syst, 2012, 28: 583–592

7 Sen J. Security and privacy issues in cloud computing. Comput Sci, 2013, 7: 238–252

8 Cao Z F. New trends of information security — how to change people's life style? Sci China Inf Sci, 2016, 59: 050106

9 Benzi M. Preconditioning techniques for large linear systems: a survey. J Comput Phys, 2002, 182: 418–477

10 Edelman A. Large dense numerical linear algebra in 1993: the parallel computing influence. Int J High Perform Comput Appl, 1993, 7: 113–128

11 Wang C, Ren K, Wang J, et al. Harnessing the cloud for securely outsourcing large-scale systems of linear equations. IEEE Trans Parall Distrib Syst, 2013, 24: 1172–1181

12 Salinas S, Luo C Q, Chen X H, et al. Efficient secure outsourcing of large-scale linear systems of equations. In: Proceedings of IEEE Conference on Computer Communications, Hong Kong, 2015. 281–292

13 Chen X F, Huang X Y, Li J, et al. New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Trans Inf Foren Secur, 2015, 10: 69–78

14 Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: outsourcing computation to untrusted workers. Lect Notes Comput Sci, 2010, 6223: 465–482

15 Yao A C. Protocols for secure computations. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science. Washington: IEEE, 1982. 160–164

16 Gentry C. Fully homomorphic encryption using ideal lattices. In: Proceedings of ACM Symposium on Theory of Computing, Bethesda, 2009. 169–178

17 Atallah M J, Frikken K B. Securely outsourcing linear algebra computations. In: Proceedings of ACM Symposium on Information, Computer and Communications Security, Beijing, 2010. 48–59

18 Nassar M, Erradi A, Malluhi Q M. Practical and secure outsourcing of matrix computations to the cloud. In: Proceedings of the 33rd International Conference on Distributed Computing Systems, Pennsylvania, 2013. 70–75

19 Li D M, Dong X L, Cao Z F. Secure and privacy-preserving pattern matching in outsourced computing. Secur Commun Netw, 2016, 9: 3444–3451

20 Lei X Y, Liao X F, Huang T W, et al. Outsourcing large matrix inversion computation to a public cloud. IEEE Trans Cloud Comput, 2013, 1: 78–87

21 Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Proceedings of International Conference on Theory and Application of Cryptographic Techniques, Prague, 1999. 223–238

22 Bellare M. Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of ACM Conference on Computer and Communications Security, Virginia, 1993. 62–73

23 Goldwasser S, Micali S, Rivest R L. A digital signature scheme secure against adaptive chosen-message attacks. Soc Ind Appl Math, 1988, 17: 281–308

24 Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM, 1978, 21: 120–126

25 Williams H. A modification of the RSA public-key encryption procedure. IEEE Trans Inf Theory, 1980, 26: 726–729

26 Blum L. A simple unpredictable pseudo-random number generator. SIAM J Comput, 1986, 15: 364–383

27 Hohenberger S, Lysyanskaya A. How to securely outsource cryptographic computations. Theory Cryptogr, 2005, 3378: 264–282

28 Liu X M, Deng R H, Ding W X, et al. Privacy-preserving outsourced calculation on floating point numbers. IEEE Trans Inf Foren Secur, 2017, 11: 2513–2527