Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix

Xing Hu[1] and Chunming Tang[2*]

Abstract

Linear algebra plays an important role in computer science, especially in cryptography. Numerous cryptographic protocols and scientific computations are based on linear algebra, which can be reduced to some core problems, such as matrix multiplication, determinant and the characteristic polynomial of a matrix. However, it is difficult to carry out these expensive computations independently for resource-limited cloud users. Outsourced computation, a service provided by cloud computing, enables a resources-constrained client to outsource his mass computing tasks to the cloud. In this paper, we use data hiding technique to design a secure and verifiable outsourcing protocol for computing the characteristic polynomial and eigenvalues of a matrix. Our protocols achieve several desired features, such as data privacy, verifiability and efficiency. Moreover, no cryptographic assumption is needed in our protocols.

Keywords: Cloud computing; Outsourced computation; Matrix; Characteristic polynomial; Eigenvalues

Introduction

Secure outsourced computation

Outsourced computation, an important service provided by cloud computing, has become more and more popular. In the context of outsourced computation, computationally weak clients (or devices) can outsource their computation and data to a server in the cloud. While this approach has numerous advantages in cost and functionality, it is crucial that the outsourcing mechanism compromises the privacy of the outsourced data and the integrity of the computation, since the server in the cloud may not be fully-trusted. The ultimate goal in secure outsourced computation is to design protocols that minimize the computational cost of the clients, and maintain the confidentiality and integrity of its data.

Generally, an outsourcing protocol is correct if the final outputs for client are valid. An outsourcing protocol is private if it is done without revealing to the external server either the actual data or the actual answer to the computation. An outsourcing protocol is verifiable if the final outputs received from cloud server can be verified by client. An outsourcing protocol is efficient if the computational work invested by client is substantially smaller than running the computation by itself, otherwise, the client could carry out the computation itself without putting confidentiality and integrity of its data at risk.

*Correspondence: ctang@gzhu.edu.cn
[2]School of Mathematics and Information Science, Guangzhou University, 510006 Guangzhou, China
Full list of author information is available at the end of the article

Outsourced computation model

Our outsourcing protocols involve two different entities: a cloud client and an untrusted cloud server. The client has many expensive matrix calculation tasks that exceed his computational abilities. So, all the tasks are outsourced to the cloud server, which has significant computation resources to perform expensive matrix calculations.

To achieve input/output privacy, the key idea is that some local preprocessing should be done on the original problem and/or data before sending it to the cloud server, and also the client needs to do some local post-processing on the answer received from the cloud server to recover the true answer. More specifically, in our model, instead of directly sending original problem φ , we use some disguising techniques to transform original problem φ into a random problem ϕ , then outsource problem ϕ to a cloud server. The server then conducts computation to get the answer of ϕ and provides a proof that the evaluation has been carried out correctly, but it cannot derive anything of the sensitive information contained in φ from the disguised problem ϕ . After receiving the solution of ϕ from the server, the client should be able to verify the validity of the answer via the appended proof. If it's valid, he recovers the desired answer for the original problem φ.

The security threats we faces are mainly from the dishonesty of the cloud server, which can be divided into semi-honest and malicious. A semi-honest cloud server corrupted by an adversary follows the protocol with the exception that it keeps recodes of all its intermediate results, he wishes to learn more sensitive information than he should obtain from the running of the protocol. If the cloud server is maliciously corrupted, then he can deviate from the protocol's description, he also can tamper with the correct data to make the honest client calculate the wrong output as well, even suspend (or abort) the execution in any desired point in time. In this paper, we assume that the cloud server is malicious.

Outsourcing protocols for some linear algebra problems with the following properties are to be designed:

- **Correctness** Any cloud server that faithfully follows the mechanism must produce an valid output.
- **Soundness** No cloud server can generate an incorrect output that can be verified successfully by client with non-negligible probability.
- **Input/output privacy** No sensitive information from client's private data can be derived by cloud server during performing the computation.
- **Verifiability** The protocol should allow client to verify the correctness of results received from an honest server with non-negligible probability, and also to detect the wrong results received from a dishonest server with non-negligible probability.
- **Efficiency** The computational work invested by the client is substantially smaller than running the computation by itself.

Our contributions

In this paper, we focus on the question of how to securely outsource the characteristic polynomial and eigenvalues of a matrix to an untrusted server in the cloud. More specifically, we provide the first efficiently and verifiably secure outsourcing protocol for the computation of the characteristic polynomial and eigenvalues by using disguising technology. In contrast to most earlier works, our protocols achieve several desired features, such as privacy, verifiability and efficiency, and no cryptographic assumption is needed in our protocols.

Related work

In 2009, Gentry [1] firstly constructed fully homomorphic encryption scheme based on ideal lattice theory, which is a significant work. Efficiently and verifiably secure outsourced computation can be constructed for any function if efficiently fully homomorphic encryption scheme exists. However, it is impossible to construct efficiently fully homomorphic encryption scheme in recent years. Some improved fully homomorphic encryption schemes were proposed [2-8], which are far away from practically fully homomorphic encryption scheme. In 2010, Gentry [2] provided a key generation algorithm for Gentry's scheme over ideal lattices in the worst-case. Van et al. [3] described a very simple somewhat homomorphic encryption scheme and convert it into a fully homomorphic scheme. Smart and Vercauteren [4] described a variant of Gentry's scheme which has smaller key size, and can be described without resorting to lattices. Stehlé and Steinfeld [5] described two improvements to Gentry's fully homomorphic scheme based on ideal lattices and its analysis. In 2011, Brakerski and Vaikuntanathan [6] presented a somewhat homomorphic encryption scheme and transform it into a fully homomorphic encryption scheme. In 2013, Lyubashevsky et al. [7] resolved an open question about LWE and its applications which have served as the foundation for cryptographic applications. In 2014, Brakerski and Vaikuntanathan [8] presented a leveled fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption.

In 2007, Kiltz et al. [9] presented secure two-party protocols for various core problems in linear algebra, such as computing the determinant and minimal polynomial of a matrix. Their protocols were based on an algorithm by Wiedemann for "black-box linear algebra" [10] which was efficient when applied to sparse matrices. Their techniques exploited certain nice mathematical properties of linearly recurrent sequences and their relation to the minimal and characteristic polynomial of the input matrix. Nevertheless, their constructions were secure under the assumption of the existence of a homomorphic public-key encryption scheme and a secure instantiation of Yao's garbled circuit protocol. And their protocol for computing determinant and the minimal polynomial of an encrypted matrix needs $O(n^2 \log n \log |F|)$ communication complexity and $O(\log n)$ rounds respectively.

In 2008, Benjamin and Atallah [11] proposed verifiably outsourced computation protocol for expensive linear algebraic computation by using a homomorphic semantically secure encryption system. However, their design is built on the assumption of two non-colluding servers and thus vulnerable to colluding attacks. In their protocols, the computation cost for client is $O(n^2)$, where n is the size of the input matrix. Two years later, based on Shamir's secret sharing scheme, Atallah et al. [12] constructed a verifiably outsourced computation protocol for matrix multiplication, which only needs one un-trusted cloud server. In their protocols, the computation cost for client is $O(t^2n^2)$, where n is the size of matrix and t is the threshold in

Shamir's secret sharing scheme. Based on Yao's Garbled Circuits and fully homomorphic encryption scheme, verifiably non-interactive outsourced computation for any function was proposed in [13], however, client needs to do an expensive preprocessing stage, and his computation complexity is related to the size of the Boolean circuit representing function .

To construct outsourced computation protocol for solving linear equation $Ax = b$, C. Wang et al. [14] realized it by using iteration from Jacobi method and additive homomorphic encryption with semantic security. In their protocols, the computation complexity for client is $O(n)$, however, their solutions are approximate and the input matrix A is constrained. In 2011, Mohassel [15] designed non-interactive and secure protocols for delegating matrix multiplication, based on a number of encryption schemes with limited homomorphic properties where the client only needs to perform $O(n^2)$ work. But their verification algorithm works correctly with an all but negligible probability in k , the probability is over the random coins of the verification algorithm (see section4.2 in [15]).

In 2012, Fiore and Gennaro [16] presented a new protocol for publicly verifiable secure outsourcing of matrix multiplication. Their scheme was in the amortized model [13], in which the client invests a one-time expensive computation phase when storing a large matrix with the server, which was used to make the verification of matrix multiplication fast. However, their design was built upon some cryptographic assumptions, such as the co-CDH assumption and the Decision Linear assumption. And their result also relied on the use of pseudo-random functions with closed-form efficiency and bilinear maps.

In 2013, Jin et al. [17] proposed an efficient parallel algorithm to compute the eigenvalue and feature vector of matrix under cloud computing environment. When computing eigenvectors, store the Laplacian matrix on the distributed file system HDFS, use distributed Lanczos to compute and get the eigenvectors by parallel computation. The time complexity of parallel computation front k eigenvector(s) is $O(kL^{op} + k^2n)$, in which L^{op} is the time that Laplace matrix multiplies a vector.

In 2013, Huang et al. [18] suggested a framework for integrating various trust mechanisms together to reveal chains of trust in the cloud. However, in present practice, individual users may not fully regard a cloud service provider or cloud server as trust assistant, the privacy and verifiability are still critical aspects of cloud computing.

Another important way to construct secure outsourcing protocols is to use mathematical disguise (or blinding) methods. In 2002, Atallah et al. [19] investigated the outsourcing of numerical and scientific computations. Their schemes applied disguise techniques to science computational problems, which guaranteed the data security and privacy. But they did not handle the important case of verification of validity of final result. In 2005, Hohenberger et al. [20] presented practical outsource-secure scheme for modular exponentiation where the honest party may use two exponentiation programs. The exponentiation programs were un-trusted and cannot communicate with each other, after deciding on an initial strategy. In 2013, several new methods of secure outsourcing of numerical and scientific computations were proposed by Seitkulov [21]. They presented different methods of finding approximate solutions to some equations solved by an external computer. Most of their methods are verifiable.

In 2013, several verifiable and secure outsourcing protocols for matrix multiplication and matrix inversion were proposed in [22]. These protocols used disguising matrix to tackle privacy-preserving problem.

Preliminaries

In this section, we give a brief review of the secure outsourcing protocol MP introduced by Hu et al. [22].

Without loss of generality, we use the notation a_{ij} to represent the element in the (i,j) position of a matrix A. The notation F_q denotes a finite field which has q elements, and $F_q^{n\times n}$ denotes the set of $n \times n$ matrices over F_q. The elementary matrix $E(i,j(k))$ is the identity matrix I but with a randomly chose $k \in F_q$ in the (i,j) position. In what follows, we also use $\delta_{x,y}$ to denote the Kronecker delta function that equals 1 if $x = y$ and 0 if $x \neq y$.

Review of the secure outsourcing protocol MP

The verifiable and secure outsourcing protocol MP for matrix multiplication [22] is reviewed as below.

Let $M_1, M_2 \in F_q^{n\times n}$ be two matrices. Suppose a client wants to obtain M_1M_2. The cloud server is not allowed to learn any sensitive information such as the input matrices, the actual output. This protocol proceeds as follows:

Step 1 The client generates a secret value for verification. He randomly picks two vectors $(a_{i1}, \ldots, a_{in})$ and $(b_{1j}, \ldots, b_{nj})^T$, which corresponding to the i-th row of M_1 and the j-th column of M_2 respectively, to computes the secret value $c = \sum_{t=1}^{n} a_{it} b_{tj}$.

Step 2 The client proceeds with the following sub-steps:

(a) Generates six private random permutations $\pi_1, \ldots, \pi_6$, where $\pi_i \in \{1, \ldots, n\}$, $i = 1, 2, \ldots, 6$.
(b) Randomly picks $6n$ non-zero numbers $\{a_1, \ldots, a_n\}$, $\{b_1, \ldots, b_n\}$, $\{c_1, \ldots, c_n\}$, $\{d_1, \ldots, d_n\}$, $\{e_1, \ldots, e_n\}$, $\{f_1, \ldots, f_n\}$ from F_q.
(c) Generates six matrices $P_1, \ldots, P_6$ where $P_1(i,j) = a_i\delta_{\pi_1(i),j}$, $P_2(i,j) = b_i\delta_{\pi_2(i),j}$, $P_3(i,j) = c_i\delta_{\pi_3(i),j}$,

$P_4(i,j) = d_i\delta_{\pi_4(i),j}, P_5(i,j) = e_i\delta_{\pi_5(i),j}$, $P_6(i,j) = f_i\delta_{\pi_6(i),j}$. These matrices are readily invertible, e.g. $P_1^{-1}(i,j) = a_j^{-1}\delta_{\pi_1^{-1}(i),j}$.

Step 3 The client computes 4 matrices

$$X_1 = P_1M_1P_2^{-1},\ Y_1 = P_2M_2P_3^{-1},$$
$$X_2 = P_4M_1P_5^{-1},\ Y_2 = P_5M_2P_6^{-1}.$$

and then sends to the server two pairs (X_1, Y_1) and (X_2, Y_2).

Step 4 The sever computes $Z_1 = X_1Y_1, Z_2 = X_2Y_2$ and return the computed output (Z_1, Z_2) to the client.

Step 5 The client recover the result by computing $T_1 = P_1^{-1}Z_1P_3$ and $T_2 = P_4^{-1}Z_2P_6$.

Step 6 After recovering, the client needs to verify the result T_1 and T_2. That is, if $T_1 = T_2$ and $T_k(i,j) = c\ (k = 1,2)$ hold simultaneously, which means the server is honest, the client obtain the correct output $P_1^{-1}Z_1P_3$ which actually equals to M_1M_2 , otherwise, the client refuses the received outputs and aborts.

Remark. *The server gains no information about the input/output matrix during the execution of the protocol. Moreover, because the matrix $P_t(t = 1, \ldots, 6)$ is special, which has only n elements rather than the matrix computed by the server which has n^2 elements, the computation time in the client side is less than the computing time in the cloud side. Recall that in [22], the protocol MP has been proved to be an efficient and verifiable outsource-secure implementation of matrix multiplication in the single cloud server model.*

Our proposed protocol

For a $n \times n$ matrix A, the computation cost for a client to compute the characteristic polynomial $f_A(\lambda)$ of A independently is proportional to n^4, and the cost for eigenvalues is at lease $O(n^3)$. If n is large (e.g. $n > 4$), it is more difficult for the client to compute the solutions of $f_A(\lambda) = 0$ independently.

We are now ready to describe our scheme. The verifiable and secure outsourcing protocol we proposed to compute the characteristic polynomial and eigenvalues of matrix works as follows. Because the probability of the real random matrix being nonsingular is 1(see Corollary 1.1 in [23]), so we assume that all eigenvalues of a matrix are nonzero.

Problem ME: let $A \in F_q^{n\times n}$ be a private matrix. Suppose a client needs to calculate the characteristic polynomial of A and its eigenvalues with the help of an un-trusted cloud server without revealing any private information, such as the input matrix, the actual output.

Protocol ME: The outsourcing protocol ME proceeded as follows:

Step 1 The client randomly picks a secret number $r \in F_q$ and then computes $A_1 = rA$.

Step 2 For each $i \in \{i, \ldots, n\}$, the client picks a random number $1 \le j \le n, j \ne i$, and then computes a matrix $B(\lambda) = L(A_1 - \lambda I)$, where $L = \prod_{i=1}^{n} E(i, j(1))$. Obviously, L is readily invertible.

Step 3 The client randomly picks a secret number $1 \le i \le n$, then divides each element $b_{ij}(\lambda)$ in the i-th row (or the i-th column) of $B(\lambda)$ to m $(1 < m < n - 2)$ random pieces. So, the client has $|B(\lambda)| = |B_1(\lambda)| + \cdots + |B_m(\lambda)|$.

Step 4 The client randomly picks two secret numbers $1 \le i, j \le n, i \ne j$, then involves the protocol MP to obtain $B_i(\lambda) \cdot B_j(\lambda)$, which denoted by $B_{m+1}(\lambda)$.

Step 5 The client hides matrices $B(\lambda)$ and $B_i(\lambda)$ $(i = 1, \ldots m + 1)$ by the following sub-steps:

(a) Generates $2m + 4$ matrices $P_1, \ldots, P_{2m+4}$. P_k is a $n \times n$ matrix for all $k = 1, \ldots 2m+4$, and $P_k(i,j) = a_i^k\delta_{\pi_k(i),j}$, where $\pi_k \in \{1, \ldots, n\}$ are random permutations, and $\{a_1^k, \ldots, a_n^k\}$ are random numbers over F_q. Let $P = \{P_1, \ldots, P_{2m+4}\}$.

(b) For each $0 \le s \le m + 1$, the client picks two matrices $P_l, P_r \in P$ from P, sets $P_l = P_l^s, P_r = P_r^s$ and $B_0(\lambda) = B(\lambda)$, then computes matrices $C_s(\lambda) = P_l^sB_s(\lambda)P_r^s$. Note that for each $B_s(\lambda)$, the disguising matrix set $\{P_l^s, P_r^s\}$ is different from each other.

(c) The client sends $C_0(\lambda), \ldots, C_{m+1}(\lambda)$ to the server. Note that the sending order of these matrices $C_s(\lambda)$ is random.

Step 6 The server computes $|C_0(\lambda)|, \ldots, |C_{m+1}(\lambda)|$ and solves $|C_s(\lambda)| = 0$ for each $0 \le s \le m + 1$. Let $\lambda^s = (\lambda_1^s, \ldots, \lambda_n^s)$ be the root vector of $|C_s(\lambda)| = 0$. Then the server return $m + 2$ two-tuples $(|C_s(\lambda)|, \lambda^s)$ to the client.

Step 7 After receiving the result from the server, the client checks the following three equalities:

$$\frac{|C_0(\lambda)|}{|P_l^0P_r^0|} = \sum_{s=1}^{m}\frac{|C_s(\lambda)|}{|P_l^sP_r^s|} \tag{1}$$

$$\frac{|C_{m+1}(\lambda)|}{|P_l^{m+1}P_r^{m+1}|} = \frac{|C_i(\lambda)|}{|P_l^iP_r^i|} \times \frac{|C_j(\lambda)|}{|P_l^jP_r^j|} \tag{2}$$

$$\prod_{j=1}^{n}\left(\lambda - \lambda_j^0\right) = |C_0(\lambda)| \tag{3}$$

If the above three equalities hold simultaneously, the client gets $|A_1 - \lambda E| = |B(\lambda)|$ from $|C_0(\lambda)|$.

Otherwise, the client refuses all answers and aborts.

Step 8 Let $g_B(\lambda) = |B(\lambda)|$, the client computes the characteristic polynomial $f_A(\lambda) = \frac{1}{r^n} g_B(r\lambda)$, and the corresponding eigenvalues $\lambda_j^* = \frac{1}{r}\lambda_j^0$ $(j = 1, \ldots, n)$.

Theorem 1. *In the single server model, protocol ME is a verifiable outsource-secure computation of the characteristic polynomial of a matrix.*

Proof.

- **Correctness:** The correctness property is straight-forward. Obviously, the server's output can be accepted successfully by the client if it is honest.
- **Soundness:** The server may act dishonestly in two ways. He may return some randomly chosen values instead of $|C_0(\lambda)|, \ldots, |C_{m+1}(\lambda)|$ or the real root vector. In the above two cases, it is infeasible for the server to satisfy the three verification equalities simultaneously in step7. The server can pass the verification with the probability at most $\left(\frac{1}{m+2}\right)\left(\frac{1}{m+1}\right)\left(\frac{1}{C_m^2}\right)\left(\frac{1}{q^m}\right)$. That is, any incorrect output generated by the server can be detected successfully by the client with non-negligible probability.
- **Input/output privacy:** The outsourcing protocol ME does not disclose the input matrix and the real result. For one thing, the client conducts a pre-disguising in step1 and step2, so the real eigenvalues and all the elements of the input matrix A (including the zero elements) are effectually hidden. For another, the second-round disguising has been done in step5. If an attacker wants to deriver any $B_s(\lambda)$ from $C_s(\lambda)$, he has to guess two permutations (from the $(n!)^2$ possible such choices) and $2n$ random numbers before he can determine a $B_s(\lambda)$. That is, for each permutation, the probability is $\frac{1}{n!}$, and for each random number, the probability is $\frac{1}{q}$, so the total probability for the attacker to obtain a correct $B_s(\lambda)$ is $\left(\frac{1}{n!}\right)^2\left(\frac{1}{q^n}\right)^2$, which is negligible.
- **Verifiability:** Assume that the server is corrupted by a PPT adversary, who wants to cheat the client without being caught. He wants to use some wrong data to make the three equalities hold simultaneously. However, to make equality (1) and (2) hold simultaneously, he has to find out the correct $|C_0(\lambda)|, |C_i(\lambda)|, |C_j(\lambda)|$ and $|C_{m+1}(\lambda)|$ from $m+2$ values $|C_0(\lambda)|, \ldots, |C_{m+1}(\lambda)|$ and properly picks m numbers α_s $(s = 1, \ldots, m)$ from F_q. Hence, the chance of the server successfully pass equality (1) and (2) is at most $\left(\frac{1}{m+2}\right)\left(\frac{1}{m+1}\right)\left(\frac{1}{C_m^2}\right)\left(\frac{1}{q^m}\right)$. Moreover, if the attacker returns to the client a randomly chosen vector instead of the right one, it is infeasible for him to make equality (3) hold for the reason that the probability of $|C_0(\lambda)|$ found from $|C_0(\lambda)|, \ldots, |C_{m+1}(\lambda)|$ is $\frac{1}{m+2}$. Therefore, the server's dishonest behavior can be caught by the client with non-negligible probability.

□

Theorem 2. *In the single server model, the protocol ME is efficient.*

Proof. The protocol ME achieves not only privacy but also efficiency. In the presented protocol, a client requires only $O(n^2)$ multiplications to compute the characteristic polynomial and eigenvalues. Concretely, the disguise conducted in step1 requires $O(n^2)$ local computation, while in step2 incurs close-to-zero additional cost on the client side. To hide matrices $B(\lambda)$ and $B_i(\lambda)$ $(i = 1, \ldots, m+1)$ in step5, the client needs $O\left((m+2)n^2\right)$ local computation, which could be small if we choose m $(1 < m < n-2)$ properly. In the verification phase (i.e. step7), the client also needs $O(n^2)$ local computation to verify equality $\prod_{j=1}^{n}\left(\lambda - \lambda_j^0\right) = |C_0(\lambda)|$. In addition, the secure outsourcing protocol MP is involved to obtain $B_{m+1}(\lambda)$ which requires $O(n^2)$ local computation in [22]. □

Conclusion and future directions

In this paper, we presented an efficient protocol for resource-limited cloud users to securely outsource the computation of characteristic polynomial and eigenvalues of matrix under cloud computing environment. In comparison, the issues studied in this paper focus on the desired features of outsourcing protocols, including privacy, verifiability and efficiency, which are critical to the availability but yet to be thoroughly studied. We also analyzed and proved the security and efficiency of our protocol in our outsourced model.

Our work leaves an open interesting problem. In this work, we design protocol for matrix whose elements are over finite field F_q. In the case that the entries are over real number field R, not every element in R is integer. So, it would be interesting to develop a verifiable computation protocol over real number field R.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

XH designed the proposed scheme for secure outsourced computation of the characteristic polynomial and eigenvalues of a matrix and drafted this manuscript. CT helped in its design and revise the manuscript and made many suggestions. Both authors read and approved the final manuscript.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No. 11271003, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant

No.20134410110003, High Level Talents Project of Guangdong, Guangdong Provincial Natural Science Foundation under Grant No.S2012010009950, the Project of Department of Education of Guangdong Province under Grant No 2013KJCX0146, and the Natural Science Foundation of Bureau of Education of Guangzhou under Grant No. 2012A004.

Author details
[1]School of Mathematics and Computational Science, Hunan University of Science and Technology, 411201 Hunan, China. [2]School of Mathematics and Information Science, Guangzhou University, 510006 Guangzhou, China.

Received: 2 November 2014 Accepted: 18 March 2015
Published online: 11 April 2015

References

1. Gentry C (2009) Fully homomorphic encryption using ideal lattices. InSTOC 9:169–178
2. Gentry C (2010) Toward basing fully homomorphic encryption on worst-case hardness. In: Advances in Cryptology-CRYPTO. Springer Berlin, Heidelberg. pp 116–137
3. Van Dijk M, Gentry C, Halevi S, Vaikuntanathan V (2010) Fully homomorphic encryption over the integers. In: Advances in Cryptology-EUROCRYPT. Springer Berlin, Heidelberg. pp 24–43
4. Smart NP, Vercauteren F (2010) Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Public Key Cryptography-PKC 2010. Springer Berlin, Heidelberg. pp 420–443
5. Stehlé D, Steinfeld R (2010) Faster fully homomorphic encryption. In: Advances in Cryptology-ASIACRYPT 2010. Springer Berlin, Heidelberg. pp 377–394
6. Brakerski Z, Vaikuntanathan V (2011) Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Advances in Cryptology-CRYPTO 2011. Springer Berlin, Heidelberg. pp 505–524
7. Lyubashevsky V, Peikert C, Regev O (2013) On ideal lattices and learning with errors over rings. J ACM (JACM) 60(6):43
8. Brakerski Z, Vaikuntanathan V (2014) Efficient fully homomorphic encryption from (standard) LWE. SIAM J Comput 43(2):831–871
9. Kiltz E, Mohassel P, Weinreb E, Franklin M (2007) Secure linear algebra using linearly recurrent sequences. Springer Berlin, Heidelberg
10. Wiedemann D (1986) Solving sparse linear equations over finite fields. Information Theory. IEEE Trans 32(1):54–62
11. Benjamin D, Atallah MJ (2008) Private and cheating-free outsourcing of algebraic computations. In: Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on. IEEE, Fredericton, New Brunswick, Canada. pp 240–245
12. Atallah MJ, Frikken KB (2010) Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security. ACM, Beijing, China. pp 48–59
13. Gennaro R, Gentry C, Parno B (2010) Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Advances in Cryptology-CRYPTO 2010. Springer Berlin, Heidelberg. pp 465–482
14. Wang C, Ren K, Wang J, Urs KMR (2011) Harnessing the cloud for securely solving large-scale systems of linear equations. In: Distributed Computing Systems (ICDCS) 2011 31st International Conference on. IEEE, Minneapolis, Minnesota, USA. pp 549–558
15. Mohassel P (2011) Efficient and Secure Delegation of Linear Algebra. IACR Cryptology ePrint Archive:605
16. Fiore D, Gennaro R (2012) Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: Proceedings of the 2012 ACM conference on Computer and communications security. ACM, Raleigh, NC, USA. pp 501–512
17. Jin R, Kou C, Liu R, Li Y (2013) Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment. J Cloud Comput: Advances Syst Appl 2(1):18
18. Huang J, Nicol DM (2013) Trust mechanisms for cloud computing. J Cloud Comput 2(1):1–14
19. Atallah MJ, Pantazopoulos KN, Rice JR, Spafford EE (2002) Secure outsourcing of scientific computations. Advances Comput 54:215–272
20. Hohenberger S, Lysyanskaya A (2005) How to securely outsource cryptographic computations. In: Theory of Cryptography. Springer Berlin, Heidelberg. pp 264–282
21. Seitkulov YN (2013) New methods of secure outsourcing of scientific computations. J Supercomputing 65(1):469–482
22. Hu X, Pei D, Tang C, WONG DS (2013) Verifiable and secure outsourcing of matrix calculation and its application. SCIENTIA SINICA: Informationis (in chinese) 43(7):842–852
23. Feng X, Zhang Z (2007) The rank of a random matrix. Appl Math Comput 185(1):689–694