Searchable symmetric encryption based on the
inner product for cloud storage

Jun Yang∗, Shujuan Li, Xiaodan Yan,
Baihui Zhang and Baojiang Cui

National Engineering Laboratory for Mobile Network Security,
School of Computer Science and Technology,
Beijing University of Posts and Telecommunications,
Beijing, P.R. China
Email: junyang@bupt.edu.cn
Email: liushujuan011@163.com
Email: yanxd2015@163.com
Email: zhang_betty1126@163.com
Email: cuibj@bupt.edu.cn
∗Corresponding author

Abstract: Searchable encryption enables the data owner to store their own data
after encrypting them in the cloud. Searchable encryption also allows the client
to search over the data without leaking any information about it. In this paper,
we first introduce a searchable symmetric encryption scheme based on the inner
product: it is more efficient to compute the inner product of two vectors. In our
construction, the parties can be data owners, clients or the cloud server. Three
parties communicate with each other through the inner product to achieve the
goal that client can search the data in cloud without leaking any information
on the data the owner stored in the cloud. We then perform a security analysis
and performance evaluation, which show that our algorithm and construction are
secure and efficient.

Keywords: inner product; searchable encryption; searchable symmetric
encryption; security; the cloud server.

Reference to this paper should be made as follows: Yang, J., Li, S., Yan, X.,
Zhang, B. and Cui, B. (2018) ‘Searchable symmetric encryption based on the
inner product for cloud storage’, Int. J. Web and Grid Services, Vol. 14, No. 1,
pp.70–87.

Biographical notes: Jun Yang received PhD in Computer Science at Beijing
University of Posts and Telecommunications in China. He is an Associate
Professor at the School of Computer Science at Beijing University of Posts
and Telecommunications. His main research areas include telecommunications,
cloud computing and big data.

Shujuan Li now is studying for Master’s in Computer Technology at Beijing
University of Posts and Telecommunications in China. Her work mainly involves
cloud computing and big data security.

Xiaodan Yan now is studying for PhD in Computer Technology at Beijing
University of Posts and Telecommunications in China. His work mainly involves
cloud computing, vehicle network and big data security.
Searchable symmetric encryption

Baihui Zhang now is studying for Master’s in Computer Technology at the Beijing University of Posts and Telecommunications in China. Her work mainly involves cloud computing and big data security.

Baojiang Cui received PhD in Control Theory and Control Engineering at the Nankai University in China. He is a Professor at the School of Computer Science at Beijing University of Posts and Telecommunications. His main research areas include software security, internet of things and big data.

1 Introduction

Some typical cloud service products have been released and received increasing attention recently with the rapid development of the technology of cloud computing (Chen and Zheng, 2009; Zissis and Lekkas, 2012; Attrapadung et al., 2012), such as the cloud storage tool Dropbox, Amazon simple storage services and Windows Azure. They store data and build a virtual system environment for clients in the cloud and transfer the data to the client through the network. Storing data in the cloud is convenient and flexible; therefore, more and more clients choose to store the local data in the cloud to save local storage space. As the data stored in the cloud are out of the control of clients, the managers of the cloud server and illegal clients could attempt to obtain the information contained in the data by accessing it, which would reveal the clients’ privacy. In recent years, due to the illegal access attempted by hackers and improper operation by the cloud managers, there were many incidents related to cloud security, resulting in a large number of data leaks. A few examples are the incident of the client information leakage at Sony in 2011, the incident of user data leakage at Google in 2011, and so on. These incidents make clients more concerned about data leaks in the cloud server. In order to ensure the confidentiality of the data, more and more companies and clients choose to encrypt data and store the encrypted data in the cloud server.

However, when the client needs to find the relevant documents containing a keyword, searching the encrypted data in the cloud server becomes a problem. The simplest way is to download all the encrypted data to the local device and decrypt it and then search over the plain text. However, in this way, the local storage space should be massive, and the computing requirements on the clients’ end should also be massive. This method is not suitable for network environments with low bandwidth. Another way is to send the key and the keyword to the cloud server so that the cloud server can decrypt the encrypted data and search over the plain text. However, this method exposes the data to the cloud manager and represents a serious security threat to the data.

To solve this problem, the concept of searchable encryption (Li et al., 2015; Bösch et al., 2014; Waters et al., 2004) is proposed. Clients should encrypt the data first, and the encrypted data should be stored in the cloud server. When clients need to search for a keyword, they can send the trapdoor for the keyword to the cloud server; the cloud server would search over the encrypted documents with the trapdoor and then send the document back to the client upon successful matching. Clients only need to decrypt the returned encrypted document. With this method, the cloud manager and adversary cannot obtain
the information related to the keyword and the plain text. Generally speaking, searchable encryption schemes fall into two categories, searchable symmetric encryption (SSE) (van Liesdonk et al., 2010; Chai and Gong, 2012) and public key encryption with keyword search (Boneh et al., 2004).

In this paper, we first introduce a new SSE algorithm based on the inner product of the vectors composed of four polynomial algorithms for security parameter setup, encryption, trapdoor generation and trapdoor testing. We then analyse the efficiency of the scheme and establish its security (Song et al., 2000; Tan et al., 2005) through detailed analysis. In SSE scheme based on the inner product of a vector, we first introduce the construction techniques for the inner product. On the basis of ensuring safety, the efficiency of the scheme is improved greatly, and thus, this scheme has further been made more feasible.

The remainder of the paper is organised as follows. Section 2 presents some related work. Section 3 provides concrete SSE based on the inner product and analyses its efficiency and security in Section 4. Finally, we conclude the paper in Section 5.

2 Related work

In this section, before we introduce our algorithm, we first review some basic mathematical concepts and cryptology concepts for the purposes of this paper.

2.1 Mathematical concepts

2.1.1 System of linear equations and the rank of the matrix

In mathematics, a system of linear equations (or a linear system) is a collection of linear equations involving the same set of variables. A solution to a linear system is an assignment of numbers to the variables such that all the equations are simultaneously satisfied. The word ‘system’ indicates that the equations are to be considered collectively, rather than individually. In mathematics, the theory of linear systems is the basis for and a fundamental part of linear algebra, a subject used in most parts of modern mathematics. Computational algorithms for finding the solutions are an important part of numerical linear algebra and play a prominent role in engineering, physics, chemistry, computer science, and economics. A system of non-linear equations can often be approximated by a linear system (see linearisation), a helpful technique for a mathematical model or computer simulation of a relatively complex system. Very often, the coefficients of the equations are real or complex numbers, and the solutions are searched in the same set of numbers, but the theory and the algorithms apply for coefficients and solutions in any field.

The number (i.e. the number of linearly independent rows or columns) is simply called the rank of a matrix M. The column rank of the matrix M is the dimension of the column space of M, and the row rank of M is the dimension of the row space of M. A fundamental result in linear algebra is that the column rank and the row rank are always equal. A matrix is said to have full rank if its rank equals the largest possible rank for a matrix of the same dimensions, which is the lesser of the number of rows and columns. A matrix is said to be rank deficient if it does not have full rank.
2.1.2 Solution to a system of linear equations

In linear algebra, an augmented matrix is a matrix obtained by appending the columns of two given matrices, usually for the purpose of performing the same elementary row operations on each of the given matrices.

For a given number of unknowns, the number of solutions to a system of linear equations depends only on the rank of the matrix representing the system and the rank of the corresponding augmented matrix. Specifically, according to the Rouche-Capelli theorem, any system of linear equations is inconsistent (has no solutions) if the rank of the augmented matrix is greater than the rank of the coefficient matrix; if, on the other hand, the ranks of these two matrices are equal, the system must have at least one solution. The solution is unique if and only if the rank equals the number of variables. Otherwise, the general solution has \( k \) free parameters, where \( k \) is the difference between the number of variables and the rank; hence, in such a case, there is an infinitude of solutions.

2.2 Broadcast encryption

The concept of broadcast encryption was first formally introduced by Fiat and Naor (1993) and has received much attention since then (Garay et al., 2000; Luby and Staddon, 1998). Broadcast encryption (Dodis and Fazio, 2003) involves a sender and a large number of users (receivers) who are listening to a broadcast channel. The sender first encrypts digital content and then transmits it to a dynamically changing set of users through insecure broadcasting channels. The users use their private keys to decrypt the information they receive. The broadcast encryption scheme (Delerable et al., 2007) assures that only privileged receivers can recover the content subscribed and that the unauthorised users can learn nothing. A broadcast encryption scheme can be described as a triple of poly-time algorithms (setup, encrypt, decrypt) as follows:

- **Setup**(\( 1^\lambda, n \)): the setup algorithm is a probabilistic algorithm used by the system to set up all the parameters of the scheme. The setup algorithm takes as input a security parameter \( 1^\lambda \) and the number of receivers \( n \) and generates the public key \( pk \) and \( n \) private keys \( p_1, \ldots, p_n \).

- **Encrypt**(\( pk, S \)): the encryption algorithm, run by the broadcaster, is a probabilistic algorithm used to encrypt a message for a subset of users. The encryption algorithm takes as input the public key \( pk \) and a subset of users \( S \subseteq \{1, \ldots, n\} \) and returns a pair \((H, K)\), where \( H \) is called the header and \( K \) is a message encryption key encapsulated in \( H \). We will often refer to \( H \) as the broadcast ciphertext. For a concrete message, it will be encrypted by \( K \) and broadcasted to the users in \( S \).

- **Decrypt**(\( pk, S, i, d_i, H \)): the decryption algorithm is a deterministic algorithm that the users run to decrypt the received messages. It takes as input a public key \( pk \), a subset of users \( S \subseteq \{1, \ldots, n\} \), a user id \( i \in \{1, \ldots, n\} \), the private key \( p_i \) for user \( i \) and a header \( H \) and outputs the message encryption key \( K \) or the failure symbol \( \bot \). \( K \) will be used to decrypt the received messages.
To ensure that the system is correct, it is required that for all $S \subseteq \{1, \cdots, n\}$ and all $i \in S$, if $(pk; (p_1, \cdots, p_n)) \xleftarrow{\$} \text{setup}(1^\lambda, n)$ and $(H, K) \xleftarrow{\$} \text{encrypt}(pk, S))$, then decrypt$(pk, S, i, d_i, H) = K$.

### 2.3 Searchable encryption

As shown in Figure 1, a searchable encryption scheme (Wang et al., 2010; Li et al., 2011) serves four processes:

**Step 1:** Encryption process. The data owners use the key to encrypt the document on the local devices and upload it to the cloud server.

**Step 2:** Trapdoor generation process. The authorised clients use the key to generate the trapdoor for a keyword, and the trapdoor cannot reveal any information on the plain text.

**Step 3:** Retrieval process. The cloud server takes as input a trapdoor to search over the ciphertext to return all of the documents containing the specified keyword. The cloud server cannot obtain more information except for that on whether the ciphertext contains a specific keyword.

**Step 4:** Decryption process. The clients use the key to decrypt the encrypted data returned by the cloud server to obtain the query results.

*Figure 1* Steps in searchable encryption (see online version for colours)

From the perspective of the key, searchable encryption schemes can be divided into two categories, SSE (Curtmola et al., 2006) and searchable asymmetric encryption (Hwang and Lee, 2007; Baek et al., 2008). The construction of symmetric encryption is usually based on the pseudo-random function. The symmetric searchable encryption algorithm, with high efficiency and low computational expense, uses the same key in the encryption and decryption process, and the same key is required for the generation of the trapdoor. The asymmetric searchable encryption (Cui et al., 2016) algorithm uses two kinds of keys: the public key, which is used to encrypt the plain text and search over the ciphertext, and the private key, which is used to decrypt the ciphertext and generate the trapdoor for the keyword. The asymmetric encryption algorithm is usually complex, and the encryption and decryption are slow.
3 Searchable symmetric encryption based on the inner product

In this section, we first describe a scenario applying the SSE scheme and then provide requirements for designing a valid SSE scheme. Finally, we propose a concrete SSE algorithm based on the inner product.

3.1 Description of the scenario

Consider a scenario such as the one shown in Figure 2, where two employees of a company are willing to share some confidential business data using public cloud (Cao et al., 2011b; Li et al., 2010) storage services, assuming that the public cloud storage is honest and curious, such as Dropbox and Amazon. For example, Alice hopes to upload a lot of financial documents to the cloud to save local storage space, and these documents are to be reviewed by different department directors. If these documents contain highly sensitive information, these documents can only be accessed by the authorised users and cannot be obtained by the public from the cloud server. Bob is one of these directors and is only authorised to view the documents related to his department. In order to ensure the security of the data in the public cloud server, Alice encrypts the file using different keys and generates keyword ciphertexts based on the department names, and she uploads the encrypted data and the encrypted keyword to the cloud. Alice then uploads and shares these documents to the directors using the cloud storage. In order to let Bob view documents related to his department, Alice should give Bob the right to search over the documents, and Bob should also have rights to decrypt these documents. In order to ensure the security of the key, Alice should securely send all the searchable encryption keys to Bob. After receiving these keys, Bob stores them securely, and he then generates the keyword trapdoor using these keys to search over the encrypted data. According to the scheme, we assume that Alice has a private document set \( \{doc_i\}_{i=1}^n \), and for each document \( doc_i \), there is a searchable encryption key \( k_i \). Generally speaking, we suppose that Alice wants to share \( m \) documents \( \{doc_i\}_{i=1}^m \) with Bob. In this case, Alice should send the searchable encryption keys \( \{k_i\}_{i=1}^m \) to Bob. Then, when Bob wants to retrieve documents containing a keyword \( w \), he should generate the keyword trapdoor \( Tr_i \) for the document \( doc_i \) with key \( k_i \) and submit the trapdoors \( \{Tr_i\}_{i=1}^m \) to the cloud server. Then, the cloud server would search over the encrypted data and return the encrypted data to the client if the query is successful. The client receiving the data decrypts it with the known \( k \).

Figure 2  The scenario where searchable symmetric encryption is applied (see online version for colours)
3.2 Requirements for designing a SSE scheme

The SSE framework provides good guidance, so that we can more easily and conveniently design a concrete algorithm based on the framework. However, in order to construct a practical scheme, we should consider more aspects of the implementation of the scheme, including function implementation, execution efficiency and query privacy.

- Function encryption: Based on SSE, data owners encrypt the data and upload the encrypted data to the cloud server; the data owners distribute different keys to different authorised (Shi and Waters, 2008) users. On this basis, we require the concrete algorithm to encrypt the keyword. Users generate the trapdoor with the received key and upload the trapdoor to the cloud server. The specific method for generating the trapdoor should be considered carefully. How to search (Shi et al., 2007; Boneh and Waters, 2007) over the encrypted data with the trapdoor submitted by the user in the cloud is the most important issue, and the whole process can be realised currently.

- Execution efficiency: For constructing the practical scheme, the efficiency (Ballard et al., 2005) should be considered throughout the process. For application in practice, the efficiency of each part of the algorithm should be considered. In general, both the efficiency and the security cannot be met simultaneously. The efficiency is low while the security is high, and the efficiency is high when the security cannot be met; therefore, we should find a balance between two points in the construction of the scheme.

- Query privacy: In a SSE scheme, we should not only ensure the security (Golle et al., 2004; Chang and Mitzenmacher, 2005; Yang et al., 2006) of the encrypted data but also consider the privacy (Cao et al., 2011a) of the trapdoor. First, to ensure the security of the encrypted data, we need to improve the security (Shen et al., 2009) of the encryption algorithm, so that the adversary cannot detect the information related to the keyword from its ciphertext. Second, users generate the trapdoor and upload it to the cloud (Dong et al., 2008); thus, the privacy (Bao et al., 2008) of the trapdoor should be considered. After the users generate the trapdoor for a keyword, the adversary cannot obtain the information of the key and the keyword in the case that the adversary obtains a number of trapdoors.

3.3 The proposed scheme

3.3.1 Overview

The inspiration for designing the SSE scheme based on the inner product (Okamoto and Takashima, 2009; Katz et al., 2013) comes from both the searchable encryption scheme and the controlled functional encryption scheme. Specifically, in the SSE scheme, based on the inner product, we use the method of the inner product of two vectors in the Controlled Functional Encryption (CFE) scheme (Naveed et al., 2014). Suppose there are two n-dimensional vectors $V \{x_1, x_2, \cdots, x_n\}$ and $K \{y_1, y_2, \cdots, y_n\}$, the inner product of these two vectors can be expressed as $<V, K> = x_1y_1 + x_2y_2 + \cdots + x_ny_n$. In the SSE scheme, the trapdoor generation algorithm run by the clients and the search algorithm run by the cloud server are all based on the inner product of two vectors. The calculation of
the inner product of two vectors is simple and efficient, and the security can be attributed to the problem of solving linear equations.

3.3.2 Description of the scheme

Based on the general idea of the searchable encryption scheme described in Section 2.3, we propose a concrete SSE scheme that contains the following algorithms, and we use graphics to describe the algorithm in a simple manner, as shown in Figure 3.

- **Setup(1^λ)**: the algorithm, run by the data owner, is a probabilistic key generation algorithm. It takes as input a security parameter \( \lambda \) and returns a secret key \( K \). The data owner runs the algorithm while generating a secret key \( K \) corresponding to a document. The secret key \( K \) of the SSE scheme should be distributed to the authorised users.

- **Encrypt(\( K, D \))**: The algorithm, run by the data owner, is used to encrypt the document and generate its keyword ciphertexts. It takes as input the secret key \( K \) and the data \( D \) of a document and outputs the encrypted data and the keyword ciphertexts by computing \( C_W = W + K \). The data owner uploads both the encrypted data and keyword ciphertexts to the cloud server.

- **Trapdoor(\( K, W \))**: The algorithm, run by the clients, is a probabilistic algorithm used to generate a trapdoor for a keyword to perform a keyword search. It takes as input the secret key \( K \) distributed to the particular client and a keyword \( W \) and outputs the trapdoor for the keyword. The client selects the vector \( V \) with the same dimensions as that of the vector \( K \) randomly and computes the inner product of the two vectors \( V, K, < V, K > \) and the inner product \( < V, R > \). We can then generate the trapdoor \( Tr = ( < V, K >, < V, W >, V ) \). The client should upload \( Tr \) to the cloud server.

- **Search(\( Tr, C_W \))**: The algorithm, run by the cloud server, is a deterministic algorithm used to search over the encrypted data to justify whether the encrypted document contains the specific keyword. It takes as input the trapdoor \( Tr \) and the keyword ciphertext \( C_W \) and computes the inner product \( < V, C_W > \), and it then judges \( < V, K > + < V, W > = < V, C_W > \). If it is true, the cloud server would send the encrypted document to the client; otherwise, the cloud server would output false.

**Figure 3** The concrete SSE scheme based on the inner product (see online version for colours)
4 Security analysis and the performance evaluation of the SSE scheme based on the inner product

4.1 Security analysis

In order to analyse the security of the scheme, we assume that the cloud server is ‘curious and honest’, and we assume that the users try to access data within or out of the scopes of their privileges. Moreover, we assume that the communication channel is not safe, that is, an attacker can obtain information from the communication channel.

The Scheme provides provable secrecy for encryption, in the sense that the attacker given the ciphertext cannot obtain any information on the plain text. The scheme provides the function of controlled searching, and illegal users cannot search for a keyword without authorisation from the data owner. The scheme realises the privacy of the query, and the authorised user can query the document without revealing the word to the cloud server.

**Theorem 1:** The proposed scheme realises the function of controlled searching.

**Proof:** The theorem requires that users who have the searchable encryption key $K$ can search over the specific document, but the users are unable to search over documents out of the scopes of their privileges. Similarly, the user cannot generate other keys to search for other files based on the known key. Theorem 1 can be deduced from the following lemmas:

**Lemma 1:** Every authorised user can perform a keyword search successfully.

**Proof:** Lemma 1 shows the correctness of the proposed scheme. After the data owner has generated the searchable encryption key, the key should be sent to the authorised user. The authorised user uses the key to generate the trapdoor, and the trapdoor would be submitted to the cloud server. The cloud server would run the search algorithm after receiving the trapdoor. We can see that the authorised user can search successfully based on the known $K$ by computing $< V, K > + < V, W > = < V, C_W >$.

**Lemma 2:** When a user tries to search for a document out of the scopes of his privilege, or when the cloud server colludes with the user, they are unable to search for documents that they are not allowed to search.

**Proof:** The proof of the lemma can be attributed to the solution of linear algebraic equations. Given $m$ equations and $n$ unknowns, $x_1, x_2, \ldots, x_n$ and $n - m \geq 80$, the solution space of the equation can be at least $2^{80}$ if the free variables $x_{m+1}, x_{m+2}, \ldots, x_n \in \{0, 1\}$ according to Section 2.1.2.

In this case, the malicious user has not only his own secret key but also information on the cloud server: trapdoor $Tr$ and the keyword ciphertext $C_W$. However, the user cannot search for documents out of the scope of his or her privileges. The malicious user has an arbitrary amount of information: $< V, K >, < V, W >, < V, W + K >$ and $V$. We assume that a document has $m$ keywords and that the malicious user knows the $m$ inner product $< V_1, K >, < V_2, K >, \ldots, < V_m, K >$ and $m$ vectors $V_1, V_2, \ldots, V_m$. According to the solution of the linear equation, the user cannot guess the value of $K, W$ or $W + K$ if $n - m \geq 80$. A malicious user generates the error $K$, selects the vector $V$ with the same
dimension as that of vector $K$, and computes the value $< V, K >, < V, W >$, which would be uploaded to the cloud server. The cloud server runs the search algorithm. We can see that there is no inner product $< V, W + K >$ to make the formula $< V, K > + < V, W > = < V, C_W >$ compute successfully, and the search algorithm would return a false value.

**Lemma 3:** The user cannot generate the new key for any other documents based on the known key to search for other documents.

**Proof:** The malicious user tries to generate a new key $R$ according to the known key $K$. According to the description of the SSE scheme, the key $K$ is generated randomly, and we guarantee that its dimension is greater than or equal to 80. If $K \in \{0, 1\}$, its value space is $2^{80}$, so the malicious user cannot guess the value of other $K$ values. Based on the above-mentioned fact, the malicious user computes the value $< V, K >$ with the false $K$ and uploads it to the cloud server. After receiving the trapdoor $Tr$, the cloud server computes $< V, K > + < V, W > = < V, C_W >$. There is no inner product $< V, W + K >$ to make the formula $< V, K > + < V, W > = < V, C_W >$ compute successfully, and therefore, the search algorithm would output false.

**Theorem 2:** The scheme realises the function of query privacy.

**Proof:** The malicious user knows the key $K$ corresponding to the specific document and can search for the documents within the scope of his key. The cloud server also knows the information on the data stored in it: the keyword ciphertext $C_W$, the trapdoor $Tr$ and the vector $V$ generated by the user randomly. However, we can still prove that our scheme achieves the function of query privacy. Theorem 2 can be deduced from the following lemma.

**Lemma 4:** An attacker cannot obtain the information on the keyword according to the known trapdoor.

**Proof:** We assume that an attacker $A$ has an arbitrary number of trapdoors, and the attacker tries to obtain the information on the keyword based on this. In this case, the attacker has the information: $< V_1, W_1 >, < V_2, W_2 >, \ldots , < V_n, W_n >$ and $V_1, V_2, \ldots , V_n$. We can describe this simply as follows: an attacker knows the inner product $< V, W >$ and the vector $V$ and tries to determine the value of $W$. According to the proof of Lemma 2, the attacker can only have a negligible probability of obtaining it.

**Lemma 5:** The cloud server cannot get the information on the keyword when the cloud server knows the keyword ciphertext and other related public information.

**Proof:** The cloud server attempts to obtain the information on the keyword from the keyword ciphertext stored in the cloud server. The cloud server knows the following information: the keyword ciphertext $C_W$ and the trapdoor $Tr$. First, for the case where the cloud server knows $C_W$, because the key $K$ is selected randomly and its value is arbitrary, the cloud server can only have a negligible probability of obtaining the keyword. Second, according to the proofs of Lemmas 2 and 4, the cloud cannot determine the key $K$ and the keyword $W$ based on the known trapdoor $Tr$. Thus, we can prove that an attacker cannot
get information on the keywords when the attacker knows all the data stored on the cloud server.

4.2 Performance evaluation

According to the SSE scheme based on the inner product described above, we can see that the algorithms in the scheme mainly involve the operation of vector addition and vector inner product. First, we perform experiments to compare the running time of these two operations. Second, considering the input time of the users’ data in practical applications, we perform experiments to test the total time taken by the operations and in generating the vectors.

4.2.1 Inner product computation and additive computation

In our experiments, we use the language Java to implement the program. The dimension of the vector is 80, and the data type used is integer. We tested our construction on a Lenovo computer with an Intel(R) Core(TM) i5-4590 CPU at 3.30 GHZ and 8 GB of RAM running Windows7. We present experimental results on the inner product operation of the vectors as follows. As shown in Figure 5, when the number of vectors increases to 100,000, the time cost is 9 ms; when the number of vectors increases to 1 million, the time cost is 47 ms; and when the number of vectors increases to 5 million, the time cost is 200 ms. Therefore, we can draw two conclusions: the operation speed for calculating the inner product is fast enough, and the time cost of the operation of the inner product computation is linear in the number of vectors.

For the operation of vector addition, the experimental results are shown in Figure 4. When the number of vectors increases to 10,000, the time cost is 2 ms; when the number of vectors increases to 1 million, the time cost is 16 ms; and when the number of vectors increases to 5 million, the time cost is 78 ms. Thus, we can draw the following conclusions: the operation of vector addition is faster than the operation of the inner product computation, and with an increasing number of vectors, the gap between the times increases. The time cost of the operation of vector addition is linear in the number of vectors.

Figure 4 Time cost of the operation of vector addition (see online version for colours)
In addition, considering the input time of the users’ data in practical applications, we performed experiments to test the total time taken by the operations and in generating the vectors. The experimental results are presented in Figures 6 and 7. We can see that the efficiency is obviously decreased when the time for generating the random vector is included, and the proportion of the time taken is about 97%, as is seen by comparing Figures 4, 5, 6 and 7. Figures 6 and 7 present almost the same results, that is, the total time based on vector addition is almost same as the total time based on the inner product computation when the time required for generating the random vector is included. We can also see that the operation of vector addition and the operation of the inner product computation are fast.

**Figure 5**  Time cost of the operation of inner product computation (see online version for colours)

**Figure 6**  Total time cost including that required for generating the random vector based on the operation of inner product computation (see online version for colours)
Figure 7  Total time cost including that required for generating the random vector based on the operation of vector addition (see online version for colours)

Finally, we use multiple threads to carry out experiments. We carry out experiments for each of the above four cases separately when the number of threads is 2, 4, 5 and 10. The results are shown in Figures 8–11: when the number of threads is 2, the time cost is about half of the time required for a single thread; when the number of threads is 4, the cost time is about 1/4 of the time required for a single thread. In the latter case, the time cost is minimum. When the number of threads is 5 or 10, the time cost is higher than that for 4 threads because the machine we used has 4 cores. We conclude that the time cost is the minimum when the number of threads is the same as the number of cores, and it is approximately equal to the value of the time cost for a single thread divided by the number of cores.

Figure 8  Time cost of the operation of inner product computation using multiple threads (see online version for colours)
Searchable symmetric encryption

Figure 9  Time cost of the operation of vector addition using multiple threads (see online version for colours)

Figure 10  Total time cost including that required for generating the random vector based on the operation of inner product computation using multiple threads (see online version for colours)

Figure 11  Total time cost including that required for generating the random vector based on the operation of vector addition using multiple threads (see online version for colours)
4.2.2 Evaluation of the SSE algorithm based on the inner product and the performance comparison

The proposed scheme involves four algorithms: setup, encrypt, tapdoor and search. Setup generates a security parameter for a document, and we see that its execution time is linear in the number of documents. Encrypt is based on the operation of vector addition. The relationship between the time and the number of keywords is presented in Figure 4; we can see from Figure 7 that the time required for generating the random vectors is included. Figure 4 shows that when the number of keywords reaches 500 million, the time required is only 78 ms. It can be seen that the efficiency of the encryption algorithm in our scheme is very high. Trapdoor computes two inner products: \( <V, W> \) and \( <V, K> \). The relationship between the time and the number of documents is presented in Figures 5 and 6. According to Figure 5, we can conclude that the time required for computing two inner products is about 400 ms when the number of keywords reaches 500 million. Search computes the inner product \( <V, C_W> \), and the relationship between the time and the number of keyword ciphertexts is presented in Figures 5 and 6.

The SSE scheme based on the advanced encryption standard (AES), also known as Rijndael (Daemen and Rijmen, 1999), and the searchable asymmetric encryption scheme based on the pairing computation are widely used currently, and there are known conclusions about both schemes. In the same experimental environment, we use the AES algorithm in Java class library to perform the experiment and find that the time cost for encrypting the string composed of 80 numbers is 110 ms. For the pairing computation, according to the pairing-based cryptography library, the time cost of a pairing computation using the pretreatment for the parings of type ‘a’ is 11 ms. According to the java pairing-based cryptography library, the time cost is 7.234 ms. According to our experiments, the time cost of 100 inner product computations is 1 ms. Thus, from the point of view of the realisation of the program, the SSE scheme based on the inner product is more efficient.

5 Conclusion

Research on searchable encryption is becoming popular, and the technology guarantees the security of the owners’ data in the search process. Based on this concept, for the first time, we apply the operation of inner product computation to our construction and realise a concrete SSE scheme on the basis of ensuring security. Moreover, we analysed the performance and security of the concrete scheme in detail. The analysis results show that our work can provide an effective solution to searching data stored in the cloud based on the SSE scheme.

In the concrete scheme we constructed, the data owner generates keyword ciphertexts using the operations of vector addition; authorised user generates the trapdoor using the operations of inner product computation, and the cloud server searches over the encrypted data using the addition operations. The operations of the inner product computation and vector addition are fast, and therefore, the efficiency of our scheme based on these two operations is very high. However, in the scheme proposed in this paper, a different key is used for a file, and if the data owner possesses a large number of documents, a large number of keys are needed, and the data owner needs to send the keys to the authorised users. Moreover, a large number of trapdoors must be generated by users and submitted to the cloud in order to perform a keyword search over many files. How to reduce the number
of encryption keys and the number of trapdoors is the follow-up work we have to consider. Moreover, this scheme provides us a direction to solve the problem of SSE, and we can continue to study more problems to be solved in this field.

References


Searchable symmetric encryption


