Privacy-assured Boolean Adjacent Vertex Search over Encrypted Graph Data in Cloud Computing

Hong Zhu[1], Bin Wu[1], Meiyi Xie[1] and Zongmin Cui[2]
[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[2] School of Information Science and Technology, Jiujiang University, Jiujiang, China
[e-mail: wubincs@gmail.com]
*Corresponding author: Bin Wu

Received December 8, 2015; revised March 29, 2016; revised May 21, 2016; accepted August 13, 2016; published October 31, 2016

Abstract

With the popularity of cloud computing, many data owners outsource their graph data to the cloud for cost savings. The cloud server is not fully trusted and always wants to learn the owners' contents. To protect the information hiding, the graph data have to be encrypted before outsourcing to the cloud. The adjacent vertex search is a very common operation, many other operations can be built based on the adjacent vertex search. A boolean adjacent vertex search is an important basic operation, a query user can get the boolean search results. Due to the graph data being encrypted on the cloud server, a boolean adjacent vertex search is a quite difficult task. In this paper, we propose a solution to perform the boolean adjacent vertex search over encrypted graph data in cloud computing (BASG), which maintains the query tokens and search results privacy. We use the Gram-Schmidt algorithm and achieve the boolean expression search in our paper. We formally analyze the security of our scheme, and the query user can handily get the boolean search results by this scheme. The experiment results with a real graph data set demonstrate the efficiency of our scheme.

***Keywords*:** encrypted graph, boolean expression, adjacent vertex search, cloud computing, searchable encryption

1. Introduction

Data outsourcing is an important application in cloud computing, and it can reduce the overhead and expenses. Hence data owners could outsource a large number of graph data to the cloud for cost savings. A graph is particularly useful in many fields, such as network topological graph [1], collaboration network [2], and social network graph [3]. In cloud computing , security is a big concern [4][5]. Due to the cloud server being not fully trusted, the graph data are usually encrypted before outsourcing to the cloud server for preventing the information leak. However, the encrypted graph data can put operations to great trouble, and this brings query users great inconvenience. Therefore, achieving the boolean adjacent vertex search over encrypted graph data in cloud computing is a very useful operation.

The adjacent vertex search which searches for all adjacent vertices of a query vertex is a very common operation in the graph [6][7][8], many other operations such as subgraph mining [9], community finding [10], and outlier detection [11], can be built based on the adjacent vertex search. The boolean adjacent vertex search over a graph is a more general operation, which searches for all adjacent vertices by a boolean expression. There are disjunction, conjunction and negation three basic operations in boolean expression [12]. For example, in a scientific collaboration network, each vertex represents a scientific researcher. We could search for the scientific collaborators through the adjacent vertex queries. We use $\{t_1, t_2, \ldots, t_x\}$ to represent *x* scientific researchers. The disjunctive operation allows a query user to search for the neighbor vertices which are adjacent to t_1 *or* t_2 … *or* t_x. The conjunctive operation allows a query user to search for the neighbor vertices which are adjacent to "t_1 *and* t_2 … and t_x". The negative operation allows a query user to search for the neighbor vertices which are not adjacent to particular vertices. Thus, through such boolean search, a query user can perform a more general adjacent vertex search. To achieve this query, we adopt boolean expression algorithms in mathematics [12]. However, it is a challenging task to directly conduct the boolean search over encrypted graph data. Considering the "pay-for-use" billing rule in the cloud, it is particularly uneconomical to download all the graph data and decrypt locally. Therefore, it is very meaningful to perform the boolean adjacent vertex search over encrypted graph data in cloud computing.

For searching encrypted data in the cloud, searchable encryption [13][14][15][16][17] is a very useful technique. The cloud server can securely perform search operation over encrypted data through a query token. Currently, searchable encryption is a hot research direction, and there have been a lot of literatures [18][19][20][21][22] in this field. Moataz et al. [18] presented an efficient searchable encryption scheme to perform boolean expression query, and proved the scheme secure through a simulation based formal proof. The problems of dynamic searchable encryption were studied in these literatures [19][20][22]. Cash et al. studied highly-scalable searchable encryption scheme with support for boolean queries in the literature [21]. But all the searchable encryption methods cannot be directly used to conduct boolean adjacent vertex search over encrypted graph data. Meanwhile, many studies of privacy-preserving graph queries [23][24][25][26] have already appeared in recent years. The problem of secure subgraph query was studied in the literatures [23][26]. Chase et al. [24] proposed the notion of structured encryption and the application of controlled disclosure. The problem of privacy-assured substructure similarity query over encrypted graph-structured data was investigated in the literature [25]. However, the problem of boolean adjacent vertex search over encrypted graph data has not been solved.

To solve the problem, we propose a solution to perform secure boolean adjacent vertex search over encrypted graph data in cloud computing (BASG). Our BASG scheme can achieve a boolean adjacent vertex query through an index, and insure the query security. We first construct graph vertices' normalized orthogonal keyword set and adjacent vector information, and then build a secure index. Then a query user can send the boolean query token to the cloud server, and the cloud server performs computing and search operation in the index by the query token. After the retrieving, the cloud server returns the boolean search results to the query user. The experiment results and security analysis show that our scheme is efficient and provably secure.

The contribution of this paper can be summarized as follows.

(1) We propose a scheme to solve the problem of the boolean adjacent vertex search over encrypted graph data in cloud computing.

(2) We formally analyze the security of our scheme, and it ensures the privacy of the boolean query tokens and the search results.

(3) The experiment results demonstrate the efficiency of our scheme.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 introduces the basic knowledge of the paper. Section 4 provides the scheme introduction and a description of BASG. Section 5 analyzes the security of our scheme. Section 6 evaluates our scheme from the experiments. Finally, section 7 concludes the paper.

2. Related Work

There are two kinds of encryption search models: the symmetric encryption search model and the asymmetrical encryption search model [13][15][16][17]. Generally speaking, the symmetric encryption search is far more efficient than the asymmetrical encryption search, thus we use the symmetric encryption search model in our scheme designing.

The searchable encryption [13][14][15][16][17] has been widely studied as a cryptographic primitive and is very effective for achieving search on encrypted data. Song et al. [13] first proposed the notion of searchable symmetric encryption which took advantage of the stream cipher to encrypt all the words in each file. Goh [14] first introduced the notion of the secure index, which made use of a bloom filter to build an index for each file. The problem of searching on data that was encrypted using a public key system was studied in the literature [15]. Chang et al. [16] presented a definition based on simulation, and it aimed to insure the privacy of the index and encrypted query word. Curtmola et al. built non-adaptive and adaptive searchable symmetric encryption schemes respectively in the literature [17]. Recently some extended searchable encryption methods have been put forward in the literatures [18][19][20][21][22]. An efficient scheme was presented in the literature [18] to focuse on the enhancement of searchable encryption, and the scheme supported any boolean expression query over encrypted data. The construction of boolean symmetric searchable encryption in the paper is mainly based on the orthogonalization of the keyword field according to the Gram-Schmidt process. In addition, the security and the efficiency were formally analyzed. The dynamic searchable symmetric encryption scheme was proposed in the literature [19]. The dynamic symmetric searchable encryption schemes in very-large databases was designed and implemented in the literature [20]. Cash et al. [21] proposed the highly-scalable searchable symmetric encryption supporting boolean queries. A new dynamic searchable symmetric encryption scheme that achieved the highest privacy and low information leakage was developed in the literature [22]. But all the schemes cannot be directly used to perform boolean adjacent vertex search.

In recent years, some studies on privacy-preserving graph queries have appeared [23][24][25][26]. Cao et al. [23] defined and solved the problem of privacy-preserving query over encrypted graph-structured data in cloud computing. To improve efficiency, the principle of “filtering-and-verification” was utilized to prune as many negative data graphs as possible before verification, and a feature-based index was pre-built to provide feature-related information for every encrypted data graph. Then, a secure inner product computation was proposed to meet the challenge of supporting graph semantics, and a variety of privacy requirements were achieved. The notion of structured encryption and the application of controlled disclosure were introduced in the literature [24]. In this paper, the schemes were constructed for encrypting graph data queries, but did not perform more complex queries such as boolean queries and similarity queries on encrypted graph data. Zhang et al. [25] constructed a framework and a series of algorithms to solve the problem of privacy-assured substructure similarity query over encrypted graph-structured data based on the privacy homomorphism and obscuration methods. Fan et al. proposed a private subgraph query approach over outsourced graph databases in the literature [26]. One idea of this paper was to determine the minimized candidate subgraphs that contained at least a candidate mapping, and then candidate mappings were enumerated from those subgraphs instead of the original graph. In the paper, a subgraph cache was proposed to prune the candidate matchings that were enumerated, and a robust encoding scheme and its verification method were presented for the users to determine the proper encodings for their queries without leaking query structures. However, all the schemes cannot solve the problem of boolean adjacent vertex search over encrypted graph data. In our paper, we construct a complete scheme based on the searchable encryption and the Gram-Schmidt algorithm, and formally analyze the query security. We build the adjacent vector information of all the graph vertices to generate a secure index, and the cloud server can perform computing and search operation by the index and encrypted query tokens. Through our mechanism, the more complex boolean adjacent vertex search could be achieved.

3. Preliminaries

3.1 Cryptographic Basics

Goldwasser and Micali introduced the notion of semantic security in the literature [27]. Informally, a system is semantically secure if whatever an adversary can compute about the plaintext given the ciphertext, he can also compute without the ciphertext [27]. In cloud computing, we need to consider the efficiency and cost, and symmetric encryption is more efficient than asymmetric encryption [15][17][28][29][30][31]. Therefore, we use symmetric encryption in this paper. A semantically secure symmetric encryption scheme is a set (*Kge*, *Enc*, *Dec*) consisting of three polynomial time algorithms [32]. *Kge* is a secret key generating algorithm which takes a security parameter as input, and returns a secret key. *Enc* is an encryption algorithm that takes a secret key and the plaintext as inputs, and returns the ciphertext. *Dec* is a decryption algorithm that takes the ciphertext and a secret key as inputs, and returns the decrypted content.

Generally speaking, a pseudorandom function is the function that cannot be distinguished from a truly random function by any efficient procedure that can get the values of the function at arguments of its choice [32]. If the pseudorandom function is an injective function, the function is a permutation, and the pseudorandom permutation and truly random permutation are indistinguishable in polynomial time [32].

3.2 Notations and Formulas

For simplifying the following description, let $t \longleftarrow T$ denote that an element t is selected from the set T, and $t \overset{R}{\longleftarrow} T$ denote that an element t is selected from the set T uniformly and randomly [17][32]. The main notations used in this paper are listed in **Table 1**.

Table 1. Summary of notations

Notations	Denotations
G	A graph data set
I	A secure index
A	A set of all vertices in the graph G
W	A set of normalized orthogonal keywords from all vertices in the graph G
n	The number of vertices in the graph G
m	The maximum degree of the graph G
Q	A query token
R	A set of search results
$f(key, \cdot)$	A pseudorandom permutation used in our scheme designing
$Enc(key, \cdot)$	A semantic security symmetric encryption function used in our scheme designing
$Dec(key, \cdot)$	A semantic security symmetric decryption function used in our scheme designing

There are three basic boolean query operations: the disjunction, the conjunction and the negation. To build our search scheme, similar to the literature [18], we adopt the Gram-Schmidt algorithm [12][18] to achieve the boolean adjacent vertex search. In linear algebra, every spanning set of a vector space must contain at least as many elements as any linearly independent set of vectors from the vector space [12].

Theorem 1. Gram-Schmidt algorithm [12]. Let $\{\alpha_1, \alpha_2, . . . , \alpha_n\}$ be a linearly independent vectors set of a pre-Hilbert space, $\theta(\alpha_i, \alpha_j)$ denotes the scalar product of α_i and α_j , the Gram-Schmidt process works as follows:

$$\beta_1 = \alpha_1;$$

$$\beta_k = \alpha_k - \sum_{i=1}^{k-1} \frac{\theta(\alpha_k, \beta_i)}{\theta(\beta_i, \beta_i)} \cdot \beta_i, \ k = 2, \ldots, n;$$

$$\eta_k = \frac{\beta_k}{\sqrt{\theta(\beta_k, \beta_k)}}, \ k = 1, \ldots, n.$$

Thus, we obtain the normalized orthogonal set $\{\eta_1, \eta_2, \ldots, \eta_n\}$ of vectors set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$.

4. BASG Construction

4.1 Scheme Overview

In the cloud, the outsourcing system model illustrated in **Fig. 1** mainly consists of three different entities: the cloud server, the data owner, and the query users. The cloud server is “honest-but-curious”, that is, it honestly follows the protocol specification, but can curiously infer and analyze the contents on the cloud server and try to learn the additional information. The data owner can outsource the encrypted graph data and the index to the cloud server. When searching, the query user sends the search tokens to the cloud server, and the cloud server can perform retrieval through the encrypted search tokens and the secure index and return the search results to the query user. For protecting the privacy, our scheme follows the security definition of searchable encryption [17].

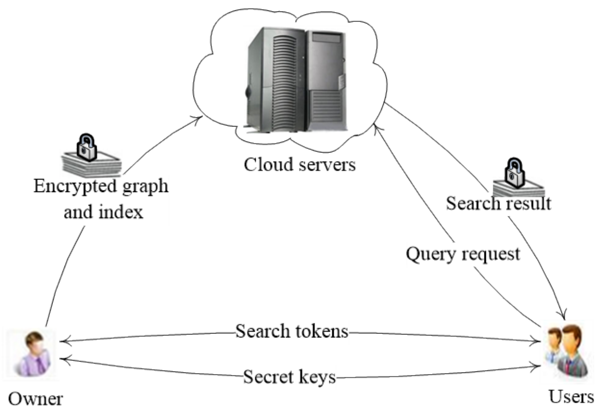

Fig. 1. Architecture of the boolean adjacent vertex search over encrypted graph

Our BASG scheme can perform the boolean adjacent vertex search over encrypted graph data in cloud computing, and should achieve the following design goals.

(1) Secure boolean adjacent vertex search functionality. The user can correctly search all the query results by a boolean token.

(2) Security. The cloud server cannot learn the contents of the query tokens and the encrypted search results.

(3) Efficiency. As little as possible computation and storage cost is spent in the search.

Some data structures such as the index, the array, and the query table are used in our BASG scheme. The data owner stores the adjacent information in the index, the query user sends the encrypted boolean search tokens to the cloud server, and the cloud server can return the search results to the query user after the retrieval. To guarantee the privacy of the stored contents and query tokens in the cloud, we must assure the security and correctness in our scheme implementation.

To attain the goal, we carry out three steps to achieve the boolean adjacent vertex search scheme. Firstly, we need to build normalized orthogonal keyword set from the graph vertex set, and construct the adjacent vertex label set of the graph vertex. Secondly, we build the adjacent vertex vector information of all the graph vertices, and generate a secure index. Finally, we construct the boolean query tokens according to the query requirements. Afterwards, the query user sends the search tokens to the cloud server, and the cloud server performs the retrieval by the query tokens along with the index and returns the query results to the user.

Consistent with most of the searchable symmetric encryption (SSE) methods [17][18][30], we assume that the adversary (i.e., the cloud server) can adopt the adaptive attack model and the data owner has the mutual request authentication and search control mechanisms e.g., broadcast encryption [17] with the query users.

KeyGen(ℓ):

Generate random secret key $K \xleftarrow{R} (0,1)^{\ell}$.

BuildOrthoKeyword(A, K):

1. The set of all the vertices in graph G is $A = \{a_1, a_2, \ldots, a_n\}$. To prevent the cloud server to learn the number of adjacent vertices, m disturbing values $\{a_{n+1}, a_{n+2}, \ldots, a_{n+m}\}$ are added into the set A, and get the new vertices set $A = \{a_1, a_2, \ldots, a_{n+m}\}$, where m is the maximum degree of the graph G.
2. Each element in the set A is transformed by pseudorandom permutation, and get the set $V = \{v_1, v_2, \ldots, v_{n+m}\}$, that is $v_i = F(K, a_i)$, and $v_i \in \{0,1\}^*$, where $1 \le i \le n+m$.
3. Build normalized orthonormal set $W = \{w_1, w_2, \ldots, w_{n+m}\}$ from the set V through theorem 1, building process is as follows.

   (1) Define temporary variables $u_1, u_2, \ldots, u_{n+m}$, get the following results:

   $u_1 = v_1$;

   $u_k = v_k - \sum_{i=1}^{k-1} \frac{\theta(v_k, u_i)}{\theta(u_i, u_i)} u_i$, $\quad k = 2, \ldots, n+m$.

   (2) Compute and get $w_k = \frac{u_k}{\sqrt{\theta(u_k, u_k)}}$, $k = 1, \ldots, n+m$.

BuildIndex(W, K):

1. Let $T = [1, n]$ denote all integers from 1 to n, let $J_i \subseteq T$ denote the label of adjacent vertices of v_i, where $1 \le i \le n$. If $|J_i| < m$, $m - |J_i|$ integers are uniformly at random from the set $\{n+1, \ldots, n+m\}$, and get a set P_i, $P_i \subset \{n+1, \ldots, n+m\}$.
2. Compute adjacent vertex vector of each graph vertex v_i, that is $N_i = \sum_{k \in J_i \cup P_i} w_k$, where $1 \le i \le n$. Thus, get the index table $I = \{N_1, N_2, \ldots, N_n\}$.

TokenGen(A, $\{\#\}$, K):

1. Let $M = \{a_1, a_2, \ldots, a_n\}$ denote a subset of set A, an arbitrary boolean expression on M can be expressed as $B(M) = a'_1 \# a'_2 \# \cdots \# a'_t$, where # is a binary logical operator, $\# \in \{\wedge, \vee\}$, $a'_i \in \{a_i, \widetilde{a_i}, \Phi\}$, $\widetilde{a_i}$ denotes the negation of a_i, Φ represents null value. For a conjunctive expression $C_e = \wedge_{j \in Y} a_j$, we define $Q_{C_e} = code(K, \wedge_{j \in Y} a_j) = \frac{1}{\sum_{j \in Y} x_j} \sum_{j \in Y} x_j w_j$, where Y is the set of vertex labels in the conjunctive expression, and $|Y|$ strictly positive integers $\{x_j\}_{1 \le j \le |Y|}$ are picked at random.
2. All boolean expressions can be expressed in disjunctive normal form, thus, the expression $B(M)$ can be transformed into $B(M) = \vee_{r=1}^{s} \wedge_{i \in T_{q_r}} a'_i$, where $T_{q_r} \subseteq T$, $1 \le r \le s$. We introduce a new vertex value λ which is the adjacent vertex of each graph vertex by default. After the Gram-Schmidt process, we have one normalized orthogonal vector γ transforming from λ. For each $r \in [1,s]$, We define: $Q_r = \frac{1}{\rho_r + \sum_{i \in L_{q_r}} \mu_{r,i}} (\rho_r \gamma + \sum_{i \in L_{q_r}} \mu_{r,i} w_i + \sum_{i \in F_{q_r}} \delta_{r,i} w_i)$,

   where ρ_r is a random positive integer, $\mu_{r,i}$ is a positive integer selected randomly, and $\delta_{r,i}$ is a random negative integer. L_{q_r} and F_{q_r} represent respectively the labels of positive and negative keywords in the boolean query expression. To avoid the particular case where there are only negative vectors in the conjunction C_r, then we introduce a vector γ in our definition. Otherwise, Q_r would have to be divided by 0.
3. After the expression transformation and coding, get the query token $Q = code(K, \vee_{r=1}^{s} \wedge_{i \in T_{q_r}} a'_i) = \begin{pmatrix} Q_1 \\ \vdots \\ Q_s \end{pmatrix}$.

Search(Q, I):

1. Perform search operation:

   For $1 \le i \le$ n:

$$\theta(Q, N_i) = \begin{pmatrix} \theta(Q_1, N_i) \\ \vdots \\ \theta(Q_s, N_i) \end{pmatrix} = \begin{pmatrix} \frac{1}{\rho_1 + \sum_{i \in L_{q_1}} \mu_{1,i}} (\rho_1 + \sum_{i \in L_{q_1} \cap D_i} \mu_{1,i} + \sum_{i \in F_{q_1} \cap D_i} \delta_{1,i}) \\ \vdots \\ \frac{1}{\rho_s + \sum_{i \in L_{q_s}} \mu_{s,i}} (\rho_s + \sum_{i \in L_{q_s} \cap D_i} \mu_{s,i} + \sum_{i \in F_{qs} \cap D_i} \delta_{s,i}) \end{pmatrix},$$

   if $\exists r \in [1, s]$, $L_{q_r} \cap D_i = D_i$ and $F_{q_r} \cap D_i$ is an empty set, then $\theta(Q_r, N_i) = 1$, the graph vertex a_i matches the query token, and we add v_i to the search results set R.
2. The search results set R is returned to the query user.

Fig. 2. Scheme designing

4.2 Scheme Designing

Our main work in our scheme is to build the secure index and the boolean query tokens. Then the cloud server can perform the search and computing. The several algorithms used in our scheme are illustrated as follows.

- *KeyGen*(1^ℓ): Takes a secure parameter ℓ as input and outputs a symmetric key K.
- *BuildOrthoKeyword*(A,K): Takes the vertex set A in the graph data G and the symmetric key K as inputs, outputs a normalized orthogonal keywords set W.
- *BuildIndex*(W,K): Takes the normalized orthogonal keywords set W and the symmetric key K as inputs, outputs a secure index I.
- *TokenGen*(W,{ #},K): Takes the normalized orthogonal keywords set W, binary logical operator set {#} and the symmetric key K as inputs, outputs a query token Q.
- *Search*(Q, I): Takes the query token Q and the index I as inputs, outputs the query results set R.

The pseudorandom permutation [32], the symmetric encryption algorithm [32][33][34][35], and the Gram-Schmidt algorithm [12][18] are used in our scheme. Let ℓ be security parameter that can be used in the key generating algorithm. Our proposed boolean adjacent vertex search scheme is described in **Fig. 2**. The building process of our scheme is as follows.

(1) Normalized Orthogonal Keyword Construction

For the outsourcing graph G, $A = \{a_1, \dots, a_n\}$ (a_i is a positive integer, and $1 \le i \le n$) is the vertices set. To hide the number of adjacent vertices for each vertex, we add m disturbing integer values $\{a_{n+1}, \dots, a_{n+m}\}$ (m is the maximum degree of the graph), and get the changed vertices set $A = \{a_1, \dots, a_{n+m}\}$. The data owner needs to build the normalized orthogonal set of the graph vertices by *BuildOrthoKeyword* algorithm. The data owner first transforms each element a_i ($1 \le i \le n+m$) via the pseudorandom permutation $f(K, \cdot)$ (K is a secret key), and get the integer value $v_i = f(K, a_i)$, that is $v_i \in \{0, 1\}^*$. The transformed set is $V = \{v_1, \dots, v_{n+m}\}$. Then, the data owner transforms the set V into the normalized orthogonal set by **theorem 1**, and the transformed vectors set is $W = \{w_1, \dots, w_{n+m}\}$.

Example 1. **Fig. 3** shows a scientific collaboration graph, each vertex represents a scientific researcher, and each edge represents the cooperative relationship between two researchers. The graph has 5 vertices and 8 edges, and the vertices set is $A = \{a_1, a_2, a_3, a_4, a_5\}$, The maximum degree of the graph is 4. we add 4 disturbing integer values $\{a_6, a_7, a_8, a_9\}$ in the set A and get the changed vertices set $A = \{a_1, \dots, a_9\}$. We perform the calculation by the pseudorandom function $v_i = f(K, a_i)$ ($1 \le i \le 9$), and the resulting values are denoted as a set $V = \{v_1, \dots, v_9\}$. The elements in the set V are transformed by theorem 1, and we get the normalized orthogonal vectors set $W = \{w_1, \dots, w_9\}$.

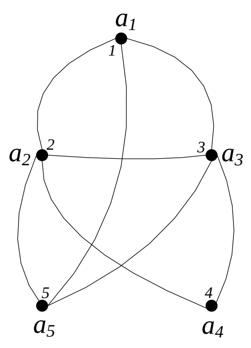

Fig. 3. A scientific collaboration graph

(2) Building Index

To achieve querying on the encrypted cloud server, the data owner needs to build the secure index by *BuildIndex* algorithm. And the cloud server can accomplish the boolean adjacent vertex search by means of the secure index. In the first place, the data owner builds the adjacent index of each graph vertex, that is, for each graph vertex a_i ($1 \le i \le n$), its all adjacent vertices label is denoted as the set J_i. The elements among the set J_i which is the subset of the set $T = [1, n]$ (T is the set of positive integers smaller than n) are positive integers. If $|J_i| < m$, $m - |J_i|$ integers are randomly chosen from the set $\{n+1, \ldots, n+m\}$, and these integers form a set P_i. In this way, when computing the adjacent vertex vector N_i, it can pad $|P_i|$ disturbing adjacent vertices. Afterwards, the data owner computes the adjacent vertex vector of each graph vertex a_i, and get the result $N_i = \sum_{k\in D_i} w_k$, where $D_i = J_i \cup P_i$. After all the computations, we have generated the index $I = \{N_1, \ldots, N_n\}$. In the end, the data owner stores the encrypted index on the cloud server.

Example 2. As mentioned above, the vertices set in **Fig. 3** is $A = \{a_1, \ldots, a_9\}$, and the normalized orthogonal vectors set of which is $W = \{w_1, \ldots, w_9\}$. we definite the label set of the vertices $\{a_1, \ldots, a_9\}$ as $\{1, \ldots, 9\}$. All the adjacent label sets of graph vertices are as follows: $D_1 = \{2, 3, 5\} \cup \{7\} = \{2, 3, 5, 7\}$, $D_2 = \{1, 3, 4, 5\}$, $D_3 = \{1, 2, 4, 5\}$, $D_4 = \{2, 3\} \cup \{6, 8\} = \{2, 3, 6, 8\}$, $D_5 = \{1, 2, 3\} \cup \{8\} = \{1, 2, 3, 8\}$, where $\{7\}$, $\{6, 8\}$ and $\{8\}$ are subsets that are randomly selected from the set $\{6, 7, 8, 9\}$. The adjacent vector of each graph vertex is computed as follows: $N_i = \sum_{k\in D_i} w_k$ ($1 \le i \le 5$), and then the search index $I = \{N_1, N_2, N_3, N_4, N_5\}$ is generated.

(3) Building Query Tokens and Performing Search Operation

In the boolean adjacent vertex search, we need to build the boolean expression query tokens according to the requirements. There are usually three kinds of operations: the disjunction operation ($\vee$), the conjunction operation ($\wedge$), and the negation operation ($\tilde{}$), where the disjunction and conjunction are binary operations. Let $M = \{a_1, \ldots, a_t\}$ be an integer set of keywords, and M is the subset of the set A. A boolean query expression on M is expressed as $B(M) = a'_1 \# a'_2 \# \ldots \# a'_t$, where # is a binary operator, and $\# \in \{\vee, \wedge\}$. And a'_i comes from the set $\{a_i, \tilde{a}_i, ø\}$, where $\tilde{a}_i$ denotes the negation of a_i, ø represents null value, and $1 \le i \le t$.

For a conjunctive expression $C_e = \wedge_{j\in Y} a_j$, where a_j is a positive keyword, we define [18]:

$$Q_{C_e} = code(K, \wedge_{j\in Y} a_j) = \frac{1}{\sum_{j\in Y} x_j} \sum_{j\in Y} x_j w_j$$

Where K and Y are respectively the secret key and the set of vertex labels in the conjunctive expression, and $|Y|$ strictly positive integers $\{x_j\}_{1\le j\le |Y|}$ are picked at random. The query user sends Q_{C_e} to the cloud server. For each adjacent vertex vector N_i ($1 \le i \le n$), the cloud server performs the following computing [18]:

$$\theta(Q_{C_e}, N_i) = \frac{1}{\sum_{j\in Y} x_j} \sum_{j\in Y} \sum_{k\in D_i} x_j \theta(w_j, w_k)$$

$$= \frac{1}{\sum_{j\in Y} x_j} \sum_{j\in Y\cap D_i} x_j$$

If $Y\cap D_i = Y$ then: $\theta(Q_{C_e}, N_i) = 1$ and N_i matches the conjunctive expression. Else no adjacent vertex vector matches the conjunctive expression.

The disjunctive (conjunctive) expression consisting only of a finite number of words is called simple disjunctive (conjunctive) expression. The disjunctive (conjunctive) expression consisting only of a finite number of simple conjunctive (disjunctive) expressions is called disjunctive (conjunctive) normal form. Any propositional formulae has an equivalent disjunctive normal form and an equivalent conjunctive normal form. In boolean logic, a disjunctive normal form is a standardization of a logical formula which is a disjunction of conjunctive clauses [12]. In our paper, we only consider our boolean query expression as disjunctive normal form: $B(M) = \overset{s}{\underset{r=1}{\vee}} \underset{i\in T_{q_r}}{\wedge} a_i'$, where $T_{q_r} \subseteq T$, $1 \le r \le s$, and s is is the number of conjunction expressions.

The correctness analysis of search operation:

We now consider each conjunction $C_r = \wedge_{r\in T_{q_r}} a_i'$ $(1 \le r \le s)$ in boolean query expression $B(M)$. We split T_{q_r} into two partitions such that: $T_{q_r} = \{L_{q_r}, F_{q_r}\}$ where L_{q_r} and F_{q_r} represent respectively the labels of positive and negative keywords in the boolean query expression [18]. In our scheme, we introduce a new vertex which is the adjacent vertex of each graph vertex by default. To this extent, we add an arbitrary value λ in the vertices set. After the Gram-Schmidt process, we have one normalized orthogonal vector γ transforming from λ. For each $r \in [1,s]$, we pick at random $|L_{q_r}|$ positive integers $\{\mu_{r,j}\}_{1\le j\le |L_{q_r}|}$, $|F_{q_r}|$ negative integer $\{\delta_{r,j}\}_{1\le j\le |F_{q_r}|}$, and one random positive integer ρ_r. we define [18]:

$$Q_r = \frac{1}{\rho_r + \sum_{i\in L_{q_r}} \mu_{r,i}} \left(\rho_r\gamma + \sum_{i\in L_{q_r}} \mu_{r,i} w_i + \sum_{i\in F_{q_r}} \delta_{r,i} w_i\right)$$

The reason why we have introduced a vector γ is for the case where there are only negative vectors in the conjunction C_r. Otherwise, Q_r would have to be divided by 0. Thus we get the query token:

$$Q = code(K, \overset{s}{\underset{r=1}{\vee}} \underset{i\in T_{q_r}}{\wedge} a_i') = \begin{pmatrix} Q_1 \\ . \\ . \\ . \\ Q_s \end{pmatrix}.$$

When the user sends the query token Q to the cloud server, for each adjacent vertex vector N_i, the following computations can be performed:

$$\theta(Q, N_i) = \begin{pmatrix} \theta(Q_1, N_i) \\ . \\ . \\ . \\ \theta(Q_s, N_i) \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\rho_1 + \sum_{i \in L_{q_1}} \mu_{1,i}} (\rho_1 + \sum_{i \in L_{q_1} \cap D_i} \mu_{1,i} + \sum_{i \in F_{q_1} \cap D_i} \delta_{1,i}) \\ . \\ . \\ . \\ \frac{1}{\rho_s + \sum_{i \in L_{q_s}} \mu_{s,i}} (\rho_s + \sum_{i \in L_{q_s} \cap D_i} \mu_{s,i} + \sum_{i \in F_{q_s} \cap D_i} \delta_{s,i}) \end{pmatrix}.$$

Therefore, the results of $\theta(Q, N_i)$ are interpreted as follows:

- If $\exists r \in [1, s]$, $L_{q_r} \cap D_i = D_i$ and $F_{q_r} \cap D_i$ is an empty set, then $\theta(Q_r, N_i) = 1$, the graph vertex a_i matches the query token, and we add v_i to the search results set R.
- If $\forall r \in [1, s]$, $\theta(Q_r, N_i) \neq 1$, no graph vertex matches the query token.

After the retrieval, the cloud server can return the search results set R to the query user.

5. Security Analysis

In this section, we analyze the security of our proposed scheme. We first introduce some notions used in [17] and adapt them for our BASG scheme.

History: A history is an interaction between the user and the cloud server, including a graph G and a set of q boolean expression queries $\{B(M_1),\ldots, B(M_q)\}$, denoted as $H_q = (G, B(M_1),\ldots, B(M_q))$. The partial history is denoted as $H_q^t = (G, B(M_1),\ldots, B(M_t))$, where $t \leq q$.

View: Let $H_q = (G, B(M_1),\ldots, B(M_q))$ is a history over q queries, the cloud server can only see an encrypted version of the history, let $G_E = Enc(K,G)$, and $B(M_i)$ is encoded as Q_i $(1 \leq i \leq q)$, a view is denoted as $V_k(H_q) = (G_E, I, Q_1,\ldots, Q_q)$. The partial view is $V_k^t(H_q) = (G_E, I, Q_1,\ldots, Q_t)$, where $t \leq q$.

Access Pattern: Let $H_q = (G, B(M_1), \ldots, B(M_q))$ is a history over q queries, the access pattern is the tuple $R(H_q) = (R_{Q_1}, \ldots, R_{Q_q})$, where R_{Q_i} $(1 \leq i \leq q)$ is the returned search results matching to the query token Q_i.

Search Pattern: Let $H_q = (G, B(M_1), \ldots, B(M_q))$ is a history over q queries, the search pattern is a binary symmetric matrix $\prod_q$, such that $\prod_q[i,j] = 1$ if $B(M_i) = B(M_j)$, and $\prod_q[i,j] = 0$ otherwise, for $1 \leq i, j \leq q$.

***Trace*:** Let H_q= (G, $B(M_1)$,…, $B(M_q)$) is a history over q queries, the trace is the sequence $T_r(H_q)$ =(|G|, $R(H_q)$, $\prod_q$). Where |G| is the total size of the graph G, $R(H_q)$ and $\prod_q$ are access pattern and search pattern of history H_q respectively. The trace of partial history is denoted as $T_r(H_q^t)$ = (|G|, $R(H_q^t)$, $\prod_t$), where $t \leq q$.

In our scheme, the security goal is to protect user privacy by preventing the cloud server from learning information of the graph data, the index and the query tokens. Here we adapt the simulation-based security model of [17], and prove our scheme meets the adaptive semantic security. The adaptive attack model considers the adversary (i.e., the cloud server) can choose the search request based on the search tokens and search outcomes of previous searches [17].

For the security guarantee, we follow the security definition used in the previous SSE schemes [17]. And our security strength is captured by the statement that given two histories with the identical trace, the cloud server cannot distinguish the views of the two histories [16][17]. In other words, the cloud server is not able to learn additional knowledge beyond the contents we are ready to leak (i.e., the trace) and thus our scheme is secure. The security theorem for our boolean adjacent vertex search scheme is stated below.

Theorem 2. Our boolean adjacent vertex search scheme meets adaptive semantic security.

Proof. We are going to describe a polynomial-size simulator S. For all $q \in N$, given only the trace of a partial history, the randomly chosen key K, given trace $T_r(H_q^t)$ for all $0 \leq t \leq q$, the simulator S can generate a view $(V_q^t)^*$ such that $(V_q^t)^*$ is computationally indistinguishable from the adversary's view $V_k^t(H_q)$.

For t = 0, the simulator S constructs the index I^* = { N_1^*, …, N_n^* } on the partial history $T_r(H_q^0)$, where $N_i^* \overset{R}{\longleftarrow} \{0,1\}^m$ (m represents the modulus of vector of graph vertex) for all 1 ≤ $i \leq n$. And the simulator generates at random $G_E^* \overset{R}{\longleftarrow} \{0,1\}^{|G_E|}$. The simulator S keeps a copy of I^* to be able to simulate future partial views for 1 ≤ t ≤ q. It is obvious that I^* is indistinguishable from I, and the encrypted graph G_E^* in $(V_q^t)^*$ is indistinguishable from G_E in $V_k^t(H_q)$, otherwise one could either distinguish between the output of the pseudorandom permutation and a random string of the same size, or between the output of a semantically secure symmetric encryption scheme and a random string of the same size. Thus, $(V_q^0)^*$ is indistinguishable from $V_k^0(H_q)$.

For $1 \leq t \leq q$, $T_r(H_q^t)$ contains the search pattern matrix $\prod_t$ for t queries. The simulator S will construct the query tokens (Q^*_1,…, Q^*_t) included in $(V_q^t)^*$. In the query tokens construction, the simulator S reuses the tokens (Q^*_1,…, Q^*_{t-1}) that were included in $(V_q^{t-1})^*$, and we assume that the simulator S remembers $(V_q^{t-1})^*$ and can reuse the query information in it, alternatively, S can reconstruct these query tokens from $T_r(H_q^{t-1})$.

To construct Q^*_t, the simulator S first checks if H_q^{t-1} contains $B(M_t)$ by checking if $\prod_t[t,j]$ = 1 for any $1 \leq j \leq t$ - 1. If H_q^{t-1} cannot contain $B(M_t)$, Let R_{Q_t} be the vertex set of $B(M_t)$, for all

$1 \le i \le |R_{Q_t}|$, S builds an association between each entry of R_{Q_t} and a value of generated I^* at the previous phase such that $((R_{Q_t})_i, N_i^*)$ are pairwise distinct. The simulator S creates $Q^*_t \xleftarrow{R} \{0,1\}^m \times \{0,1\}^y$ (y represents the number of disjunctions of the simulator's guess) such that for all $1 \le i \le |R_{Q_t}|$ there exists at least one $1 \le x \le y$ such that the inner product $\theta(Q^*_{t,x}, N_i^*) = 1$. Otherwise, if H_q^{t-1} contains $B(M_t)$, then the simulator S retrieves the query information associated with $B(M_t)$ and assigns it to Q^*_t. This makes sure that if H_q^t contains repeated boolean expression queries, then the query information included in $(V_q^t)^*$ are identical.

It's easy to see that the query tokens $(Q^*_1,\ldots, Q^*_t)$ in $(V_q^t)^*$ are indistinguishable from the query tokens $(Q_1,\ldots, Q_t)$ in $V_k^t(H_q)$, otherwise one could distinguish between the output of the pseudorandom permutation and a random string of the same size. For all $0 \le t \le q$, we claim that there is no probabilistic polynomial-size adversary can distinguish between $(V_q^t)^*$ and $V_k^t(H_q)$. Thus, we have proven the theorem.

6. Experimental Evaluations

In this section, we demonstrate a thorough experiment evaluation of our scheme on the Enron email network graph [36][37]. Our experiment is using C programming language on both a local machine and a cloud server. The local machine has an Intel Core 2 CPU running at 2.4 GHz, and 4GB of RAM. The cloud server is equipped with 4 CPU cores (2 × 2.8 GHz) and 8GB of RAM. In our experiment the dominant factor affecting the performance is the search phase as it is performed by the cloud server for the query users. The performance of the index construction, query tokens generation, and decryption actions in our scheme are evaluated on the client side.

In our experiment evaluation, we consider two cases about the graph data picked randomly from real network graph set, one contains more edges (shorter form GME), the other contains less edges (shorter form GLE) which have about half of the number of the former's edges. To facilitate comparison, we take the same boolean query in all these searches, and do not take into account the query construction time or the network communication overheads.

6.1 Building Index

In our index construction scheme, we first need to build the normalized orthogonal vectors of graph vertices by Gram-Schmidt algorithm. And then we construct the adjacency vector of each graph vertex and the secure index of our scheme.

The results of normalized orthogonal vectors construction can be seen in **Fig. 4**. The *Y*-axis of **Fig. 4** represents the runtime of normalized orthogonal vectors construction. The *X*-axis of **Fig. 4** shows the number of vertices. As can be seen in the figure, the normalized orthogonal vectors construction time is almost linear in the number of vertices in the graph.

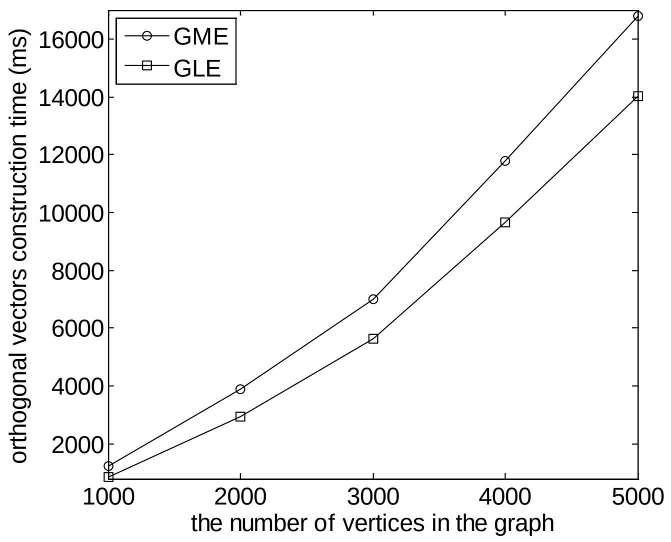

Fig. 4. Orthogonal vectors construction time

The index construction process includes orthogonal vectors building and secure index constructing, and the orthogonal vectors building is related to the number of vertices. The index construction results are plotted in **Fig. 5**, where the *Y*-axis shows the runtime of index construction, and the *X*-axis represents the number of vertices in the graph. As can be seen in **Fig. 5**, the index construction time varies almost linearly with the number of vertices in the graph. Generally, the graph that contains more edges should possess more adjacent vertices, and the index construction time of GME is more than that of GLE. Similarly, as shown in **Fig. 6**, the size of index construction is nearly linear to the number of vertices, and the index size of GLE is less than that of GME.

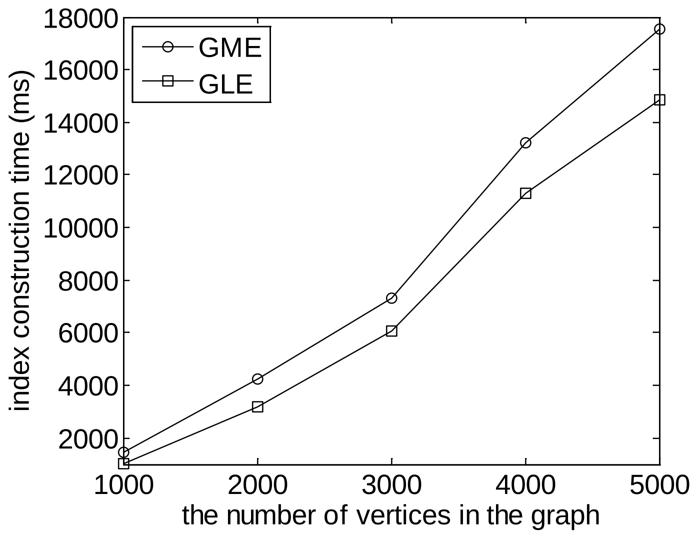

Fig. 5. Index construction time

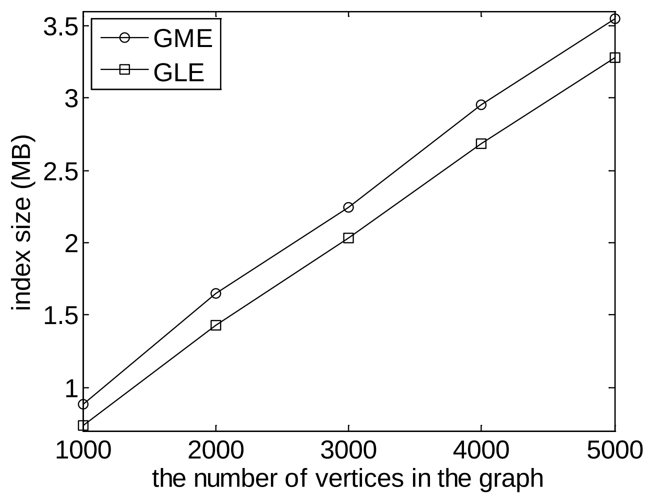

Fig. 6. Index construction size

6.2 Performing Query

In the query operation, the cloud server executes the retrieving process by the encrypted query tokens and the search index. The query process includes the retrieving on server side and the decryption of search results. The retrieving results are plotted in **Fig. 7**, where the *Y*-axis represents the query time, and the *X*-axis shows the number of vertices in the graph.

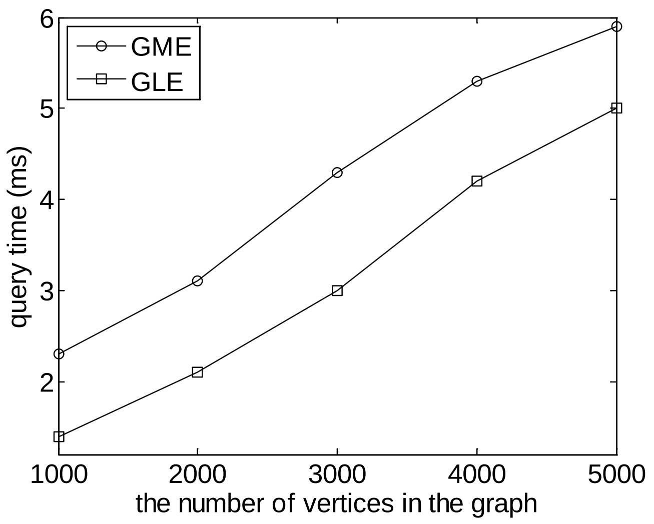

Fig. 7. Query time

As shown in **Fig. 7**, the query time is almost linear to the number of vertices in the graph, and the query of GLE has the higher efficiency than that of GME. This shows that the query will cost more time with the increasing number of edges in the graph under the same vertices.

After retrieving, the cloud server returns the encrypted search results to the query user, and the user will decrypt the results using the keys. The decryption process is illustrated in **Fig. 8**, where the *Y*-axis shows decryption time, and the *X*-axis represents the number of vertices in the graph. **Fig. 8** shows the time cost to decrypt the search results. The decryption operation is mainly determined by the size of the search results, and the decryption process is closely related to the number of vertices and edges of the graph.

As can be seen from **Fig. 5** and **6**, the index construction cost is acceptable given that these computations and operations are performed on the client side. Considering much more complex queries can be handled, the search cost on the server side would be acceptable.

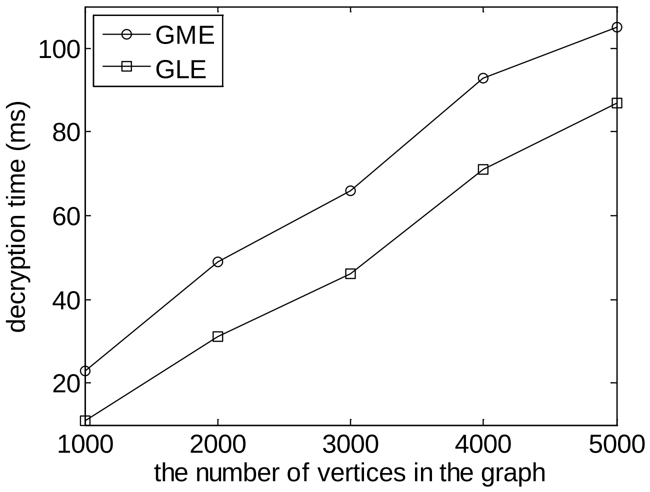

Fig. 8. Decryption tim

7. Conclusion

In this paper, we propose a solution to achieve the secure boolean adjacent vertex search over encrypted graph data in cloud computing. As the cloud server cannot be fully trusted, to accomplish the boolean adjacent vertex search, we make use of searchable encryption and the Gram-Schmidt algorithm to construct our scheme. With the implementation of our mechanism, the query users can achieve the more complex boolean adjacent vertex search. We formally prove the security in the paper. The experiment results with a real graph data set demonstrate efficiency of our scheme.

As our future work, we aim to build a dynamic boolean adjacent vertex search scheme which could support the operations of dynamic graph and meet the requirement of scalability. Another interesting direction is to build an efficient scheme which meets key management and update in an asymmetric setting.

Acknowledgement

The authors gratefully acknowledge the reviewers' comments and helpful suggestions.

References

[1] B. Zong, R. Raghavendra, M. Srivatsa, X. Yan, A.K. Singh and K.W. Lee, “Cloud service placement via subgraph matching,” in *Proc. of IEEE 30th International Conference on Data Engineering (ICDE)*, pp. 832-843, 2014. Article (CrossRef Link)

[2] T.K. Saha, B. Zhang and M.A. Hasan, “Name disambiguation from link data in a collaboration graph using temporal and topological features,” *Social Network Analysis and Mining*, vol. 5, no. 1, pp. 1-14, 2015. Article (CrossRef Link)

[3] R. West, H.S. Paskov, J. Leskovec, and C. Potts, “Exploiting social network structure for person-to-person sentiment analysis,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 297-310, 2014.
[4] N.D. Han, L. Han, D.M. Tuan, H.P. In and M. Jo, “A scheme for data confidentiality in Cloud-assisted Wireless Body Area Networks,” *Information Sciences*, vol. 284, pp. 157-166, 2014. Article (CrossRef Link)
[5] H. Li, F. Li, C. Song, and Y. Yan, “Towards Smart Card Based Mutual Authentication Schemes in Cloud Computing,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 9, no. 7, pp. 2719-2735, 2015. Article (CrossRef Link)
[6] M. Potamias, F. Bonchi, A. Gionis and G. Kollios, “k-Nearest neighbors in uncertain graphs,” in *Proc. of the VLDB Endowment*, vol. 3, no. 1, pp. 997-1008, 2010. Article (CrossRef Link)
[7] H. Maserrat and J. Pei, “Neighbor query friendly compression of social networks,” in *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 533-542, 2010. Article (CrossRef Link)
[8] M. Radovanovic, A. Nanopoulos and M. Ivanovic, “Reverse nearest neighbors in unsupervised distance-based outlier detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369-1382, 2015. Article (CrossRef Link)
[9] M.A. Bhuiyan and M.A. Hasan, “An iterative mapreduce based frequent subgraph mining algorithm,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 608-620, 2015. Article (CrossRef Link)
[10] D. Chen, Y. Dong, X. Huang, H. Chen and D. Wang, “A community finding method for weighted dynamic online social network based on user behavior,” *International Journal of Distributed Sensor Networks*, vol. 2015, 2015. Article (CrossRef Link)
[11] H.B.M. Shashikala, R. George and K.A. Shujaee, “Outlier detection in network data using thebetweenness centrality,” in *Proc. of SoutheastCon 2015*, pp. 1-5, 2015. Article (CrossRef Link)
[12] G. Strang, “*Introduction to Linear Algebra, Fourth Edition*,” Wellesley Cambridge Press, 2009.
[13] D.X. Song, D. Wagner and A. Perrig, “Practical techniques for searches on encrypted data,” in *Proc. of IEEE Symposium on Security and Privacy*, pp. 44-55, 2000. Article (CrossRef Link)
[14] E.J. Goh, “Secure indexes,” *IACR Cryptology ePrint Archive*, vol. 2003, pp. 216, 2003.
[15] J. Baek, R. Safavi-Naini and W. Susilo, “Public key encryption with keyword search revisited,” in *Proc. of Computational Science and Its Applications - ICCSA 2008, International Conference, PartI*, pp. 1249-1259, 2008. Article (CrossRef Link)
[16] Y.C. Chang and M. Mitzenmacher, “Privacy preserving keyword searches on remote encrypted data,” in *Proc. of Applied Cryptography and Network Security, Third International Conference (ACNS)*, pp. 442-455, 2005. Article (CrossRef Link)
[17] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” in *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS)*, pp. 79-88, 2006. Article (CrossRef Link)
[18] T. Moataz and A. Shikfa, “Boolean symmetric searchable encryption,” in *8th ACM Sympo-sium on Information, Computer and Communications Security (ASIACCS)*, pp. 265-276, 2013. Article (CrossRef Link)
[19] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proc. of the ACM Conference on Computer and Communications Security, CCS'12*, pp. 965-976, 2012. Article (CrossRef Link)
[20] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.C. Rosu, and M. Steiner, “Dynamic searchable encryption in very-large databases: Data structures and implementation,” in *Proc. of 21st Annual Network and Distributed System Security Symposium, NDSS*, 2014. Article (CrossRef Link)
[21] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.C. Rosu, and M. Steiner, “Highly-scalable searchable symmetric encryption with support for boolean queries,” in *Proc. of Advances in Cryptology, CRYPTO 2013 - 33rd Annual Cryptology Conference*, pp. 353-373, 2013. Article (CrossRef Link)

[22] A.A. Yavuz and J. Guajardo, "Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware," *IACR Cryptology ePrint Archive*, vol. 2015, pp. 107, 2015. Article (CrossRef Link)

[23] N. Cao, Z. Yang, C. Wang, K. Ren and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Proc. of 2011 International Conference on Distributed Computing Systems (ICDCS)*, pp. 393-402, 2011. Article (CrossRef Link)

[24] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. of Advances in Cryptology, ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 577-594, 2010. Article (CrossRef Link)

[25] Y. Zhang, S. Su, Y. Wang, W. Chen and F. Yang, "Privacy-assured substructure similarity query over encrypted graph-structured data in cloud," *Security and Communication Networks*, vol. 7, no. 11, pp. 1933-1944, 2014. Article (CrossRef Link)

[26] Z. Fan, B. Choi, J. Xu and S.S. Bhowmick, "Asymmetric structure-preserving subgraph queries for large graphs," in *Proc. of 31st IEEE International Conference on Data Engineering, ICDE 2015*, pp. 339-350, 2015. Article (CrossRef Link)

[27] S. Goldwasser and S. Micali, "Probabilistic encryption," Journal of computer and system sciences, vol. 28, no. 2, pp. 270-299, 1984. Article (CrossRef Link)

[28] Z. Cui, H. Zhu and L. Chi, "Lightweight key management on sensitive data in the cloud," *Security and Communication Networks*, vol. 6, no. 10, pp. 1290-1299, 2013. Article (CrossRef Link)

[29] Y.J. Ren, J. Shen, J. Wang, J. Han and S.Y. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, vol. 16, no. 2, pp. 317-323, 2015. Article (CrossRef Link)

[30] Z. Fu, X. Sun, Q. Liu, L. Zhou and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. 98-B, no. 1, pp. 190-200, 2015. Article (CrossRef Link)

[31] Y. Liu, H.L. Wu and C.C. Chang, "A Fast and Secure Scheme for Data Outsourcing in the Cloud," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 8, no. 8, pp. 2708-2721, 2014. Article (CrossRef Link)

[32] O. Goldreich, "*Foundations of Cryptography*," Cambridge University Press, 2004. Article (CrossRef Link)

[33] Z. Cui, H. Zhu, J. Shi, L. Chi and K. Yan, "Lightweight management of authorization update on cloud data," in *Proc. of 19th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2013*, pp. 456-461, 2013. Article (CrossRef Link)

[34] Z. Xia, X. Wang, X. Sun and Q. Wang, "A Secure and Dynamic Multi-keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340-352, 2015. Article (CrossRef Link)

[35] Z. Fu, K. Ren, J. Shu, X. Sun and F. Huang, "Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement," *IEEE Transactions on Parallel and Distributed Systems*, 2015. Article (CrossRef Link)

[36] B. Klimt and Y. Yang, "Introducing the enron corpus," in *Proc. of CEAS 2004 - First Conference on Email and Anti-Spam*, 2004.

[37] J. Leskovec, K.J. Lang, A. Dasgupta and M.W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29-123, 2009. Article (CrossRef Link)

Hong Zhu received her B.S. degree, M.S. degree and Ph.D. degree from Huazhong University of Science and Technology in 1987, 1990 and 2001, respectively. Currently, she is a professor and a Ph.D. supervisor in School of Computer Science and Technology, Huazhong University of Science and Technology in China. Her main research areas are big data management technology, big data security technology, database theory and implementation technology, and database security technology.

Bin Wu received his B.S. degree from Shandong Normal University in 2003 and the M.S. degree from Dalian University of Technology in 2009. Currently he is a Ph.D. student of School of Computer Science and Technology in Huazhong University of Science and Technology. His research interests include database security, network security, and cloud security.

Meiyi Xie received her B.S. degree, M.S. degree and Ph.D. degree from Huazhong University of Science and Technology in 1996, 1999 and 2009, respectively. Currently, she is a lecturer in School of Computer Science and Technology, Huazhong University of Science and Technology in China. Her main research areas are modern database theory and technology, database security, and cloud security.

Zongmin Cui received the B.E. degree from Southeast University in 2002 and the M.S. degree from Huazhong University of Science and Technology in 2006. He received the Ph.D. Degree from Huazhong University of Science and Technology in 2014. He is currently an associate professor with the School of Information Science and Technology, Jiujiang University. His research interests include cloud security, data query and publish/subscribe system.