Received August 11, 2020, accepted November 17, 2020, date of publication November 26, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040781

Privacy-Protection Path Finding Supporting the Ranked Order on Encrypted Graph in Big Data Environment

BIN WU[1], XIANYI CHEN[2], CAICAI ZHANG[1], ZHUOLIN MEI[1], ZHIQIANG ZHAO[1], ZONGDA WU[3], AND TAO YAN[4]

[1]School of Information Science and Technology, Jiujiang University, Jiujiang 332005, China
[2]School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
[3]School of Mathematical Information, Shaoxing University, Shaoxing 312000, China
[4]School of Information Engineering, Putian University, Putian 351100, China

Corresponding author: Bin Wu (wubcst@163.com)

This work was supported in part by the Nature Science Foundation of China under Grant 61762055, Grant 61962029, Grant 61662039, and Grant 61741111; in part by the Jiangxi Provincial Natural Science Foundation of China under Grant 20181BAB202014, Grant 20202ACBL202005, Grant 20181BAB202011, and Grant 20202BAB212006; in part by the Key Scientific and Technological Research Project of Jiangxi Provincial Education Department of China under Grant GJJ190899; in part by the Jiangxi Key Natural Science Foundation under Grant 20192ACBL20031; in part by the Science and Technology Research Project of Jiangxi Education Department under Grant GJJ180904; in part by the Humanities and Social Sciences Foundation of Colleges and Universities in Jiangxi Province under Grant TQ18111; in part by the Key Program of Zhejiang Provincial Natural Science Foundation of China under Grant LZ18F020001; in part by the Program for New Century Excellent Talents in Fujian Province University; and in part by the Natural Science Foundation of Fujian under Grant 2019J01816.

ABSTRACT Since data outsourcing in big data environment become popular and become a trend, large quantities of graph data are outsourced to the big data server for saving cost. As the big data server can not be fully reliable, people usually encrypt the graph data before they are outsourced to the big data platform for the privacy protection. The path finding is a frequently-used action and can be useful for production and living. The path finding supporting the ranked order is a more useful operation, and a user can obtain a ranked search result set. Because of the outsourced graph data being encrypted on the big data server, the path finding supporting the ranked order becomes a task with enough challenge. In this paper, we propose a solution to perform privacy-protection path finding supporting the ranked order on encrypted graph in big data environment (PPFR). Our research uses an encryption mechanism and a ranking strategy to achieve path finding supporting the ranked order. We formally analyze the security of our scheme. We demonstrate the efficiency of our proposed scheme on a real graph data set by experiment.

INDEX TERMS Path finding, encrypted graph, ranked order, searchable encryption, big data.

I. INTRODUCTION

The ubiquity of big data application promote the development of data outsourcing. Outsourcing services bring great convenience to people [1]. At present, graph is widely used in many fields, such as social network [2], road network [3], and collaboration network [4]. Considering reducing costs and expenses, data owners are willing to outsource graph data to the big data server [5]. As the big data server can not be entirely trustworthy, the outsourced graph data needs to be conducted security processing. Performing encryption processing on graph data is a very usual method before it is sent to the big data platform [6]. But it is not convenient for users to use and operate the encrypted graph data. Therefore, the implementation to perform privacy-protection path finding supporting the ranked order on encrypted graph in big data environment is a very significant job.

The path finding in the graph is a frequently-used action and can be useful for production and living [7]. Many operations and applications, such as community detection [8], outlier detection [9], and path planning [10], can be realized through path finding. The path finding supporting the ranked order is a stronger operation, and a user can obtain a ranked search result set depending on path weight. Meanwhile, to prevent the disclosure of privacy, we would process graph information and query contents through encryption

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed.

algorithm [11]. For instance, in a urban road graph, a vertex represents a location in the city, and the weight of each edge represents path length. We get the ranked information of a path based on the sum of weights between the two locations. And a user can obtain the top-k nearest distances from the query results in a path finding. A privacy-protection path finding is a path query between two vertices on encrypted graph without revealing any private information related to the query. However, it is a difficult task to conduct the path finding in the context of an encrypted graph. Considering the ''pay-on-demand'' rule in data outsourcing scenario [12], it isn't cost-effective to download all the graph data on the local machine. For that reason, it's extremely valuable to perform path finding supporting the ranked order over encrypted graph data. But it is a very challenging thing to complete the path finding in view of the privacy concerns in big data environment.

To perform search over encrypted data in big data environment, searchable encryption is a very useful method that allows the big data server run search by encrypted query token [13]–[17]. Searchable encryption is currently a research focus, and some study works have unfolded. And then a lot of researchers have proposed and implemented some searchable encryption methods that support dynamic data updates [18]–[22]. But all the methods can not be taken to execute path finding supporting the ranked order over encrypted graph. Recently, some research about encrypted graph search has emerged [23]–[27]. Chase *et al.* present the ideas of structured encryption and the using of the controlled disclosure on encrypted graph [23]. The subgraph search protecting privacy was studied by some scientific research personnel in the literature [24]–[26]. The secure reachability search problem was studied by Yin *et al.* in the literature [27]. Yet it cannot solve the problem of path finding supporting the ranked order on encryptedased on the sum of weights graph.

We solve this problem by coming up with an effective solution to implement privacy-protection path finding supporting the ranked order on encrypted graph in big data environment (PPFR). In the PPFR scheme, we achieve the secure path finding supporting the ranked order by building index, and ensure search security. We first link and encrypt any two vertices, and get the new symbols. Then we build the chained list of each new symbol including the ranked order value and path information. And then we build an index based on all the chained lists, and the index is kept on the big data server. Before excuting path finding, the query vertices are encrypted and converted to a new symbol set which is sent the big data platform. The big data server runs the path finding through the index and the query symbols, and the query results are returned to the user after query processing. The big data server cannot obtain the private information about the query results and query symbols. By the aid of security analysis and experimental results, it is showed that our PPFR scheme is provably secure and highly efficient.

The contribution of our research can be stated as follows.

(1) We design a scheme to solve the problem of path finding supporting the ranked order on encrypted graph in big data environment.

(2) We analyze the security of the scheme, and ensure the the privacy of the query process and query results.

(3) We demonstrate the efficiency of the scheme through the experiment results.

The rest of our paper is introduced below. Section II introduces the related work. Section III designs and analyzes the PPFR scheme. Section IV gives the security analysis of the PPFR scheme. Section V evaluates our PPFR scheme from the experiments. Finally, section VI summarizes our paper.

II. RELATED WORK

In the current field of information technology [28]–[31], information security is an extremely significant aspect that needs to be considered, and the protection of privacy information is an elementary need of the masses [32]–[34]. Outsourcing data security is one of our main research directions at present. Searchable encryption is of important function to querying in cloud outsourcing data [35]. A ton of data are outsourced to the remote server in the form of encryption, and the privacy-protection query can be performed over the data. In general, there has two modes about searchable encryption: symmetric searchable encryption (SSE) and asymmetric searchable encryption (ASE) [15], [16]. As symmetric encryption is more efficient than asymmetric encryption in terms of computation and overhead, symmetric encryption principle is used in our scheme design.

In outsourcing data query, searchable symmetric encryption has become a very useful basic primitive [13]–[17]. Song *et al.* first presented the idea of searchable symmetric encryption in the literature [13]. Goh adopted the bloom filter in the literature [14], and put forward the idea of the secure index to solve query question on remote server. The idea and way of non-adaptive SSE and adaptive SSE were presented by Curtmola *et al.* in the literature [16]. Chang *et al.* proposed simulation-based idea in the literature [17], and intended to protect the privacy of the search index and the query token. And then a number of expanding searchable encryption schemes were proposed in the literatures [18]–[21]. The dynamic searchable encryption method that supported the addition and deletion of outsourcing files was proposed in the literature [18]. For very-large databases, Cash *et al.* proposed and accomplished a dynamic SSE idea in the literature [19]. Hahn *et al.* presented a secure searchable encryption scheme in which the outsourcing data can be efficiently updated [20]. A SSE scheme which supported high scalability and boolean query was presented by Cash *et al.* in the literature [21]. Fu *et al.* proposed effective schemes based on concept hierarchy to address the problem of semantic retrieval in the literature [22]. But none of all the above solutions can be used to implement path finding supporting the ranked order on encrypted graph.

Some researchers studied and addressed some privacy-preserving graph query questions in recent years [23]–[27], [36]. Chase *et al.* proposed the notion of structured encryption and studied the query scheme of encrypted graph [23]. Cao *et al.* defined and addressed the problem of secure query over encrypted graph, and accomplished subgraph query through "filtering-and-verification" mechanism [24]. Zhang *et al.* used privacy homomorphism and obscuration ideas to implement privacy-preserving substructure similarity query on encrypted graph in the literature [25]. Fan *et al.* proposed a practical privacy-protection method to perform subgraph query of protection structure in the literature [26]. Yi *et al.* studied the problem of secure reachability query on outsourcing encrypted graph in the literature [27], [36]. However, existing schemes about encrytped graph queries have not solved the problem of path finding supporting the ranked order on encrypted graph.

In our paper, we think out a scheme to perform path finding supporting the ranked order in big data environment. We first build the linked lists based on the transformed graph vertices, and build an index on the basis of the linked lists, and next the queries are implemented by the index and the encrypted query symbols on the big data server. We finally give the security analysis and experiment evaluation to demonstrate the security and efficiency of our proposed scheme.

III. PPFR SCHEME CONSTRUCTION

A. PRELIMINARIES

The ideas of semantic security and indistinguishability were first proposed in the literature [37]. A scheme is semantically secure if whatever an attacker can compute about the plaintext given the ciphertext, he can also compute without the ciphertext [37]. In our design scheme, we make use of a set (*Kge*, *Enc*, *Dec*) which includes three polynomial time algorithms and is semantically secure to denote a semantically secure encryption infrastructure [38]. *Kge* is a key generating algorithm, and *Enc* denotes a symmetric encryption algorithm. *Dec* represents a symmetric decryption algorithm. The main notations used in our paper follows in Table 1.

B. DESIGN OVERVIEW

In big data outsourcing environment, the architecture of query system is illustrated in Fig. 1, and is mainly composed of three parts: the big data server, the data owner and data users. To protect the outsourced graph data and perform secure path finding, we encrypt the sensitive information of graph data and then build a query index to implement path finding on the big data server. The big data server is not trustworthy, and is responsible for storing the outsourcing graph and the index. The request of user's path finding needs to be encrypted for security reasons. In the paper, our main work focuses on the implemention of path finding supporting the ranked order on encrypted graph. For the authentication and query control of data users, we use the strategies of previous searchable encryption schemes, e.g. broadcast encryption [16], [17].

TABLE 1. Summary of notations.

Notations	Denotations
G	An outsourcing graph data set
$\mathcal{I}$	The path finding index
n	The number of vertices of the graph G
V	The outsourcing graph vertices set $V = \{v_1, \ldots, v_n\}$
m	The number of paths in the graph G
$\sharp$	String linking symbol
e	The maximum number of paths about graph vertices
$\mathcal{T}_{i,j}$	The encrypted query symbol set, where $1 \leq i, j \leq n$
$R_{\mathcal{T}_{i,j}}$	The set of path finding results of query symbol $\mathcal{T}_{i,j}$
$Enc_{key}(\cdot)$	A symmetric encryption function
$Dec_{key}(\cdot)$	A symmetric decryption function

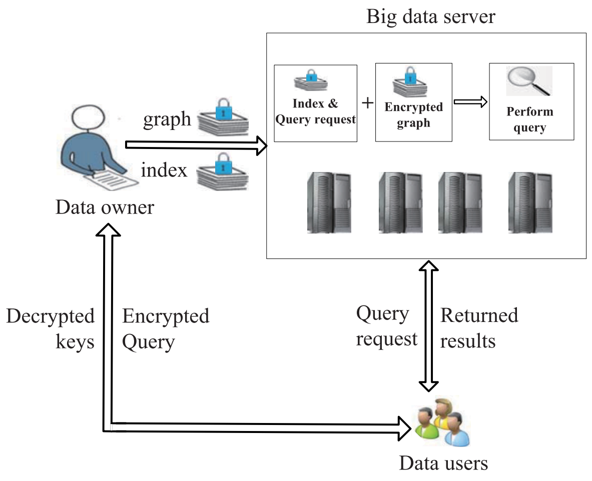


FIGURE 1. Architecture of path finding on encrypted graph.

The proposed PPFR scheme will perform path finding supporting the ranked order on the big data server, and follow the rules of the existing searchable encryption [16], [17]. Also, our scheme will achieve the following goals.

(1) Privacy-protection path finding function. A query user can achieve path finding supporting the ranked order by the encrypted query symbols.

(2) Security. The security of the proposed scheme is formally analyzed, and the privacy of query symbols and results can not be divulged to the big data server.

(3) Efficiency. Our scheme spends the less cost to accomplish the path finding functionality.

In the proposed scheme, we use several data structures such as index and chained list. In the paper, undirected or directed graphs can be used for research purposes. For the sake of uniformity, we may as well consider using the undirected graph in our design scheme. For the requirements of the ranked order, we adopt the weight calculation on the path to achieve. We use the chained list to store the encrypted graph vertices and the new symbols after conversion. The query request has to be encrypted before executing the query. To implement the secure query on the big data server, the index is needed to be built, and also it needs to prevent privacy information leaking.

To achieve the path finding in our scheme, we're going to do it in three steps. We first do the vertices processing to get new symbols and then build the path chained list of each new symbol, and each node of the chained list comprises the path relation and the ranking weight. All the nodes of each chained list are needed to be encrypted for the privacy and security. We need to build the index secondly, and the nodes of all the chained lists are randomly placed in the index. Finally, the query vertices are encrypted and processed, and then be sent to the big data server. The big data server performs path finding supporting the ranked order by encrypted symbols and the index. To be in conformity with most of the SSE ideas [16], [17], [22], we presume the big data server can use the adaptive attack model and, query users have the mutual request authentication and search control mechanisms with the data owner [16].

C. SCHEME DESIGNING

In our proposed scheme, the major problem would be to build the index and path finding method in big data environment, and several algorithms used in the scheme are listed as follows.

- *KeysG*(1^l): Keys generation algorithm. It takes l as an input parameter, and outputs a key κ.
- *Buildchainedlist*(G, $\mathcal{K}$): Building a chained list of every two vertices to load path information. It takes the graph G and the keys set $\mathcal{K}$ as inputs, and takes the set of encrypted linked symbols U and the chained list set $\mathcal{L}$ as outputs.
- *Buildindex*(U, $\mathcal{L}$, $\mathcal{K}$, $\mathcal{K}'$): Building a path finding index, and it takes the set U, the chained list $\mathcal{L}$, the key set $\mathcal{K}$, and the other key set $\mathcal{K}'$ as inputs, and the output is the index $\mathcal{I}$.
- *BuildQueryterm*(v_i, v_j, $\mathcal{K}$): Building encrypted query symbol, and the inputs are two different vertices (e.g, v_i, v_j, where, $1 \le i, j \le n$, and $i \ne j$), and the keys set $\mathcal{K}$, and the output is the encrypted query symbol set $\mathcal{T}_{v_{i,j}}$.
- *Queryindex*($\mathcal{I}$, $\mathcal{T}_{i,j}$): Excuting path finding in big data environment, and it takes the index $\mathcal{I}$ and the query symbol set $\mathcal{T}_{v_{i,j}}$ as inputs, and outputs path information set.

As mentioned above, we use (Kge, Enc, Dec) as the symmetric encryption mode in our path finding scheme, and take l as security parameter. The creating process of our PPFR scheme follows.

1) BUILDING CHAINED LIST

For the outsourcing graph G, we use $V = \{v_1, v_2, \ldots, v_n\}$ to represent the set of vertices. Every two different vertices are linked and encrypted, and a new generated symbol u_i ($0 \le i \le m$, m is the number of paths in the graph G) is got. The set of all the new symbols is represented as $U = \{u_1, u_2, \ldots, u_m\}$. Next, we're going to build a chained list $\mathcal{L}_i$ for each symbol u_i, and the specific process is shown in algorithm 1. We need to compute the sum of the weights on each path, which is used as the value for ranked order. The contents of each node in the chained list include the path information and the value for ranked order, and we uniformly use this symbol *path_cons* to express the contents of each node. To prevent privacy disclosure, we can encrypt the node contents. The two key sets used are respectively denoted as $\mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_m\}$ and $\mathcal{K}' = \{\kappa'_1, \kappa'_2, \ldots, \kappa'_m\}$. The set of all the created path chained lists is represented as $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots \mathcal{L}_m\}$. In the *Buildchainedlist* algorithm, the time complexity of building each chained list $\mathcal{L}_i (1 \le i \le m)$ is $O(max(|\mathcal{L}_i|, e) = O(e)$ (e is the maximum number of paths about graph vertices). Thus, the total time complexity of building m chained lists is $O(m \cdot e)$.

Algorithm 1 Buildchainedlist

Input: The outsourcing graph G, and the key set $\mathcal{K}$

Output: The chained list set $\mathcal{L}$

```
1:  The vertices set about graph $G$ is $V = \{v_1, v_2, \ldots, v_n\}$,
    and the used key set is $\mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_m\}$;
2:  for all $i, j \in [1, n]$ do
3:      if $i \ne j$ then
4:          The vertices $v_i$ and $v_j$ are linked and encrypted, and
            the processing result is expressed as $u_t$ $(1 \le t \le m)$.
            The all new generating symbol set is denoted as a set
            $U = \{u_1, u_2, \ldots, u_m\}$; We use $|u_t|$ to represent the
            number of paths about the symbol $u_t$.
5:      end if
6:  end for
7:  for all $i \in [1, m]$ do
8:      for all $t \in [1, |u_i|]$ do
9:          $u_{i,t} = Enc_{\kappa_i}(path\_cons)$;
10:         $\mathcal{L}_i[t] = u_{i,t}$;
11:     end for
12: end for
13: $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots \mathcal{L}_m\}$;
14: return $\mathcal{L}$.
```

2) BUILDING PATH FINDING INDEX

To implement path finding about encrypted graph in big data environment, we need to build an index so that the big data server can perform secure path finding by it. The building process of path finding index is shown in algorithm 2. For each element u_i in the set U, all of its path contents are kept in the chained list $\mathcal{L}_i$, and the number of paths is $|\mathcal{L}_i|$. For $1 \le j \le |\mathcal{L}_i|$, we build a tag about path information of u_i by linking u_i and j, and the tag is denoted as $u_i \sharp j$. And then all the tags about u_i are represented as a set $J_{u_i} = \{u_i \sharp 1, \ldots, u_i \sharp |\mathcal{L}_i|\}$. The matching path information of each element in the set J_{u_i} is stored in the path finding index $\mathcal{I}$. When carrying out path finding of u_i, it is equivalent to searching for the matching entries in the $\mathcal{I}$ by all the relational tags in the set J_{u_i}. Each tag in the set J_{u_i} corresponds to only one element in the path finding index $\mathcal{I}$. To prevent the big data server from knowing the number of paths of each element u_i, we have to add disturbing elements in the index $\mathcal{I}$, so that the number of each element's paths in the set U is the same value e, that is,

Algorithm 2 Buildindex

```
Input: $U, \mathcal{L}, \mathcal{K}, \mathcal{K}'$
Output: $\mathcal{I}$
 1: The encrypted linked symbol set is $U = \{u_1, u_2, \ldots, u_m\}$, and the generated chained list set is $\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots \mathcal{L}_m\}$; The used key sets are $\mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_m\}$ and $\mathcal{K}' = \{\kappa'_1, \kappa'_2, \ldots, \kappa'_m\}$;
 2: for all $i \in [1, m]$ do
 3:     for all $j \in [1, |\mathcal{L}_i|]$ do
 4:         $w = Enc_{\kappa'_i}(u_i \sharp j)$;
            /* w is the location of an element in the index */
 5:         $\mathcal{I}[w] = \mathcal{L}_i[j]$;
 6:     end for
 7: end for
 8: for all $i \in [1, m]$ do
 9:     if $|\mathcal{L}_i| < e$ then
10:         for all $s \in [1, e - |\mathcal{L}_i|]$ do
11:             $w = Enc_{\kappa'_i}(|\mathcal{L}_i| + s)$;
12:             $\mathcal{I}[w] = Enc_{\kappa_i}(path\_cons + s)$;
13:         end for
14:     end if
15: end for
16: return $\mathcal{I}$.
```

it is the maximum number of paths in the graph. If $|\mathcal{L}_i| < e$, we have to add $e - |\mathcal{L}_i|$ disturbing values.

In the index building algorithm *Buildindex*, for each element $\mathcal{L}_i[j]$ of each chained list $\mathcal{L}_i (1 \le i \le m, 1 \le j \le e)$, we have to calculate all the locations of the index, and have to place the contents of the index. In consequence, the time complexity of the index building is $O(m \cdot e)$.

3) BUILDING QUERY SYMBOL AND PERFORMING PATH FINDING

When the index creation is complete, the outsourcing graph and the index are stored on the big data server. For every two different vertices v_i and v_j, we build query symbol set $\mathcal{T}_{v_{i,j}} = \mathcal{T}_{u_x} = (\rho_{x1}, \ldots, \rho_{xe})$, where, $1 \le x \le m$. Specifically speaking, the query symbol set is created by a symmetric encryption function $Enc_{key}(\cdot)$. It is also to say, $\mathcal{T}_{v_{i,j}} = \mathcal{T}_{u_x} = (\rho_{x1}, \ldots, \rho_{xe}) = (Enc_{\kappa_x}(u_x \sharp 1), \ldots, Enc_{\kappa_x}(u_x \sharp e))$. When a query user is going to perform path finding about the vertices v_i and v_j, the query symbol set $\mathcal{T}_{v_{i,j}}$ (that is, $\mathcal{T}_{u_x}$) is sent to the big data server. By means of the index, the big data server will carry out path finding by Algorithm 3.

In the path finding algorithm *Queryindex*, if $\mathcal{I}[\rho_{xy}]$ is not a disturbing value for $1 \le y \le e$, we are going to add $\mathcal{I}[\rho_{xy}]$ to the path finding result set. But if the result set is a null set, no query results match the query symbol. Therefore, the time complexity of path finding is $O(e)$.

IV. SECURITY ANALYSIS

The PPFR scheme has been built, and then we are going to do a security analysis of the scheme. We first present several

Algorithm 3 Queryindex

```
Input: $\mathcal{I}, \mathcal{T}_{u_x}$
Output: $Result_{u_x}$
1: The query symbol set is $\mathcal{T}_{u_x} = (\rho_{x1}, \ldots, \rho_{xe})$, where $1 \le x \le m$;
2: for all $y \in [1, e]$ do
3:     if $\mathcal{I}[\rho_{xy}]$ is the path information then
4:         $\mathcal{I}[\rho_{xy}]$ is stored in the set $Result_{u_x}$;
5:     end if
6: end for
7: return $Result_{u_x}$.
```

notions that are used in the security analysis of our PPFR scheme [16].

• *History*: The interaction between the big data server and the path finding user, containing the outsourcing graph G and the query symbol set, represented as $H_q = (G, \mathcal{T}_1, \ldots, \mathcal{T}_q)$. The partial history is represented as $H_q^t = (G, \mathcal{T}_1, \ldots, \mathcal{T}_t)$, where $t \le q$.

• *View*: Considering a history H_q about the key κ, the view is defined as $V_\kappa(H_q) = (Enc_\kappa(G), \mathcal{I}, \mathcal{T}_1, \ldots, \mathcal{T}_q)$. The partial view is $V_\kappa^t(H_q) = (Enc_\kappa(G), \mathcal{I}, \mathcal{T}_1, \ldots, \mathcal{T}_t)$, where $t \le q$.

• *Access Pattern*: Considering a history H_q about the key κ, an access pattern is the tuple $R(H_q) = (R_{\mathcal{T}_1}, \ldots, R_{\mathcal{T}_q})$, where $R_{\mathcal{T}_i} (1 \le i \le q)$ is the path finding results matching the query symbol $\mathcal{T}_i$.

• *Search Pattern*: Considering a history H_q about the key κ, the search pattern is a binary symmetric matrix $\prod_q$, such that $\prod_q[i, j] = 1$ if $\mathcal{T}_i = \mathcal{T}_j$, and $\prod_q[i, j] = 0$ otherwise, for $1 \le i, j \le q$.

• *Trace*: Considering a history H_q about the key κ, the trace is the sequence $T_r(H_q) = (|Enc_\kappa(G)|, R(H_q), \prod_q)$, where $|Enc_\kappa(G)|$ is the overall scale of the encrypted outsourcing graph, $R(H_q)$ and $\prod_q$ are access pattern and search pattern of history H_q respectively. The trace of partial history is represented as $T_r(H_q^t) = (|Enc_\kappa(G)|, R(H_q^t), \prod_t)$, where $t \le q$.

When performing path finding, the big data server will not know the privacy information of graph data and the path finding results. In our proposed scheme, we prove the path finding meets the adaptive semantic security guarantee [16]. As for the adaptive attack model, the adversary (i.e., the big data server) can choose the query request according to the query symbols and path finding results of previous queries [16]. For the implementation of security analysis, we are in obedience to the security idea that is used in the previous scheme implementation [16], [17]. In the light of the security of our PPFR scheme, the big data server can not know the excess contents beyond the information we are willing to divulge, i.e., the trace, and hence our scheme is of security. The security theorem for our path finding is made the following statement.

Theorem 1: Our path finding supporting the ranked order on encrypted graph meets the adaptive semantic security.

Proof: To prove the security of our scheme, we first define a polynomial-scale simulator $\mathcal{S}$. For each $q \in N$, considering the trace of a partial history $T_r(H_q^t)$, the simulator $\mathcal{S}$ can generate a view $(V_q^t)^*$ which is indistinguishable from the adversary's view $V_\kappa^t(H_q)$, where κ is a randomly chosen key, and $0 \leq t \leq q$.

For $t = 0$, the simulator $\mathcal{S}$ builds the encrypted graph in $(V_q^t)^*$ by arbitrary generated string. Because of the indistinguishability principle of the semantically secure symmetric encryption $Enc_{key}(\cdot)$, the built encrypted graph in $(V_q^t)^*$ is indistinguishable from that in $V_\kappa^t(H_q)$. And then the simulator $\mathcal{S}$ builds the simulative index $\mathcal{I}^* = \{\mathcal{I}_1^*, \ldots, \mathcal{I}_m^*\}$ on the trace $T_r(H_q^0)$ of a partial history through generated random strings. The size of the index $\mathcal{I}^*$ is the same with the real index $\mathcal{I}$ which is obtained from the tace of the partial history, and the index $\mathcal{I}^*$ can be used in all partial views $(V_q^t)^*$ to simulate the adversary, where $0 \leq t \leq q$. It is quite clear that $\mathcal{I}^*$ is indistinguishable from the real $\mathcal{I}$, otherwise one can distinguish between the outputs of the symmetric encryption $Enc_{key}(\cdot)$ and the random strings of the same size. Thus, $(V_q^0)^*$ is indistinguishable from $V_K^0(H_q)$.

For $1 \leq t \leq q$, the simulative index $\mathcal{I}^*$ in the partial view $(V_q^t)^*$ still belongs to the simulator $\mathcal{S}$. $T_r(H_q^t)$ involves search pattern matrix $\prod_t$ of t path findings. The simulator $\mathcal{S}$ will construct the query symbols $(\mathcal{T}_1^*, \ldots, \mathcal{T}_t^*)$ which are included in $(V_q^t)^*$. During the build process of the query smybols, the simulator $\mathcal{S}$ will reuse the symbols $(\mathcal{T}_1^*, \ldots, \mathcal{T}_{t-1}^*)$ which were contained in $(V_q^{t-1})^*$, and alternatively the simulator $\mathcal{S}$ will reconstruct the query smybols from $T_r(H_q^{t-1})$.

To generate $\mathcal{T}_t^*$, the simulator $\mathcal{S}$ first make sure that if H_q^{t-1} involves $\mathcal{T}_t$ via checking if $\prod_t[t,j] = 1$, where $1 \leq j \leq t-1$. If H_q^{t-1} can't involve $\mathcal{T}_t$, the simulator $\mathcal{S}$ adopts the information from $T_r(H_q^t)$ about $R_{\mathcal{T}_t}$, that is $R_{\mathcal{T}_t} = R_{\mathcal{T}_{u_t}} = (R(u_t\sharp 1), \ldots, R(u_t\sharp e))$. Each $R(u_t\sharp i)$ involves only one encrypted path content, where $1 \leq i \leq e$. The simulator $\mathcal{S}$ constructs a new encrypted path content through choosing any random string. The simulator $\mathcal{S}$ randomly chooses any two addresses ad_i and ad_j from $\mathcal{I}^*$ for $1 \leq i,j \leq e$, being sure that the any two addresses are distinct, and builds the query symbol $\mathcal{T}_t^* = (ad_1, \ldots, ad_e)$. The simulator $\mathcal{S}$ remembers the correlation between $\mathcal{T}_t^*$ and u_t. Otherwise, if H_q^{t-1} involves u_t, the simulator $\mathcal{S}$ retrieves the query information corresponding to u_t and assigns it to $\mathcal{T}_t^*$. This ensures that if H_q^t involves repeated query symbols, then the query information contained in $(V_q^t)^*$ is the exact same.

It is quite obvious that the the query symbols $(\mathcal{T}_1^*, \ldots, \mathcal{T}_t^*)$ in $(V_q^t)^*$ are indistinguishable from the query symbols $(\mathcal{T}_1, \ldots, \mathcal{T}_t)$ in $V_K^t(H_q)$. Otherwise, the output of a symmetric encryption function and the random string of the same size are distinguishable. Therefore, for all $0 \leq t \leq q$, there is no probabilistic polynomial-size adversary that could distinguish between $(V_q^t)^*$ and $V_K^t(H_q)$. Thus, the security theorem of our proposed scheme have been proven. □

V. EXPERIMENTAL EVALUATIONS

For our proposed scheme, we are going to test and evaluate on the Hep-Ph citation network graph [39], [40] in this section. We carry out our experiments by using the C language program on both local working machine and big data server. Our local machine is equipped with Windows 10 system with Intel Core 4 CPU running at 2.6 GHz. The big data server is equipped with Linux system using 6 CPU cores wiht 3.0 GHz and 16 GB of RAM. In this paper, our main work is the construction of privacy-protection path finding method, including index building, query symbol building, and path finding, etc. The performances evaluations about building index, generating query symbols, and decrypting query results are made on the local machine. The path finding performance evaluation of our scheme is made on the big data server.

In general, a graph data set will have more paths if it has more edges under the same number of vertices. The different number of vertices and edges in the outsourcing graph will influence our experimental evaluations. In the experiment of our PPFR scheme, we use five outsourcing graph sets of random selection, and consider two scenarios to evaluate performance. One of the scenarios is that the outsourcing grap has more edges (marked as PPFR1), and the number of graph edges is respectively 8136, 17052, 36242, 75628, and 129573. The other scenario includes less edges (marked as PPFR2), and the number of edges is approximately half of the number in the first scenario. The comparative test between the two scenarios will be designed to assess the time and storage spendings about path finding and to validate the efficiency of our PPFR scheme.

A. INDEX BUILDING

To implement path queries and protect the security of private information on the big data server, we do this by creating secure index and encrypting query symbols. In our PPFR scheme, we first link and encrypt any two vertices in the graph vertex set to get a new symbol set. Based on the new symbol set, we build a path chained list of each new symbol through *Buildchainedlist* algorithm, and next generate secure index through *Buildindex* algorithm. After comparison of experimental evaluations on the outsourcing graph data set in the two scenarios, we give the experimental results analysis figures of index generation. The experimental results of index building time are plotted in Fig. 2, where the X-axis shows the number of vertices about the outsourcing graph, and the Y-axis represents the time of index building.

As Fig. 2 shows, we can see that the number of paths increases as the number of vertices increases in the outsourcing graph. The time of index building is nearly linear to the number of vertices in the graph. Generally, for a graph that has the same number of vertices, there are more paths if the graph contains more vertices. Consequently, the time of index building in PPFR1 scenario is more than that in PPFR2 scenario. Likewise, the analysis results about the size of index building are shown in Fig. 3. The horizontal axis in Fig. 3 shows the number of vertices in the graph, and the vertical axis shows the index size. The size of index building almost increases linearly with the number of vertices

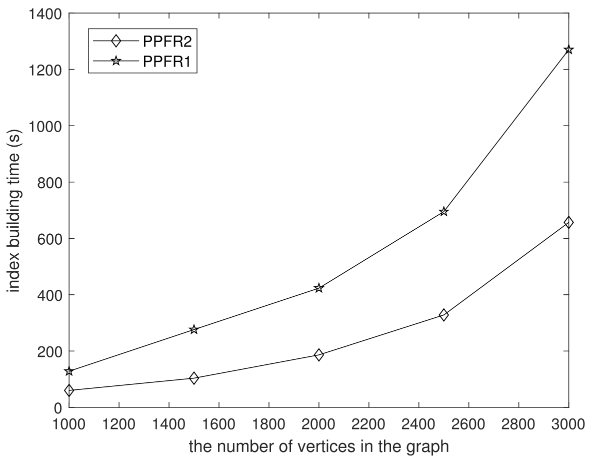


FIGURE 2. Index building time.

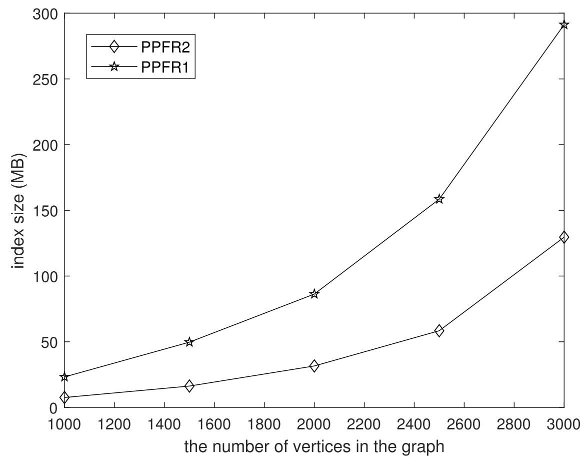


FIGURE 3. Index building size.

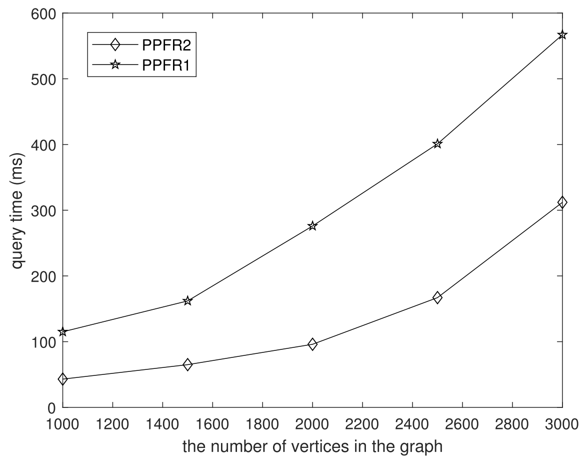


FIGURE 4. Query time.

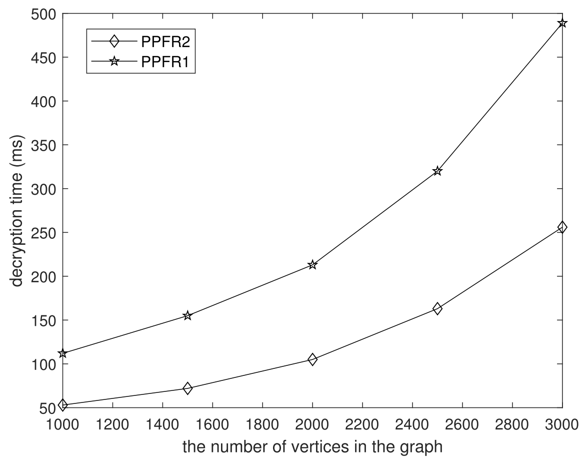


FIGURE 5. Decryption time.

increasing in the two scenarios, and the size of the building index will be the larger when there are more edges under the same number of vertices in the graph. Therefore, the size of index building in PPFR1 scenario is more than that in PPFR2 scenario.

B. PERFORMING PATH FINDING

When performing query operations, the big data server uses *Queryindex* algorithm to execute path finding by means of the index, and then the returned query results are decrypted on the local machine. The results of experimental analysis are plotted in Fig. 4 and Fig. 5, where the abscissa shows the number of vertices in outsourcing graph, and the ordinate represents the query time or decryption time. In Fig. 4, we can see that the query time about path finding is nearly linear to the number of vertices in the graph, and the query operation in PPFR2 scenario has less query time than that in PPFR1 scenario.

After the query operation about path finding is completed, the query user get the returned retrieved results from the big data server. The analysis results of decryption process in our experiment are shown in Fig. 5, where the X-axis represents the number of vertices in outsourcing graph, and the Y-axis shows the time of decryption operation. The decryption time of our scheme depends on the decryption algorithm and the size of retrieved results. The used decryption algorithms in PPFR1 and PPFR2 two scenarios are identical. As the size of retrieved results and the number of paths are identical, the decryption time and the number of paths in our scheme are interrelated. The time of decryption operation in two scenarios improves with the number of vertices improving, and is almost linear in variation trend. The time consuming of decryption operation in PPFR1 scenario is more than that in PPFR2 scenario.

In conclusion, as can be seen from our experimental analysis, the overhead of time and storage about path finding method building in our scheme almost linearly increases with the number of vertices. The operations of index building and query results decryption are done offline on the client side, and the big data server does not obtain the information of outsourcing contents, query smybols, and query results. As a result, our proposed PPFR scheme accomplishes path finding supporting the ranked order and meets security and efficiency.

VI. CONCLUSION

We propose a noval path finding scheme supporting the ranked order based on searchable encryption thought in our paper. We first give the overall design idea of our scheme which contains building chained list, generating index, and performing path finding in three steps. We next prove the security of our scheme which meets the adaptive semantic security. we finally evaluate our proposed scheme through experimental analysis, and the scheme has good performance and efficiency.

ACKNOWLEDGMENT

The authors would like to thank the editor and the reviewers' comments, suggestions and reviews.

REFERENCES

[1] W. Liao, C. Luo, S. Salinas, and P. Li, "Efficient secure outsourcing of large-scale convex separable programming for big data," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 368–378, Sep. 2019.

[2] M. A. Rogov and A. A. Sedakov, "Coordinated influence on the opinions of social network members," *Autom. Remote Control*, vol. 81, no. 3, pp. 528–547, Mar. 2020.

[3] W. Yu, Y. Zhang, T. Ai, Q. Guan, Z. Chen, and H. Li, "Road network generalization considering traffic flow patterns," *Int. J. Geographical Inf. Sci.*, vol. 34, no. 1, pp. 119–149, Jan. 2020.

[4] L. Wang, G. Li, Y. Ma, and L. Yang, "Structure properties of collaboration network with tunable clustering," *Inf. Sci.*, vol. 506, pp. 37–50, Jan. 2020.

[5] S. Fugkeaw and H. Sato, "Scalable and secure access control policy update for outsourced big data," *Future Gener. Comput. Syst.*, vol. 79, pp. 364–373, Feb. 2018.

[6] S. Tahir, L. Steponkus, S. Ruj, M. Rajarajan, and A. Sajjad, "A parallelized disjunctive query based searchable encryption scheme for big data," *Future Gener. Comput. Syst.*, vol. 109, pp. 583–592, Aug. 2020.

[7] G. A. Montoya and Y. Donoso, "A prediction algorithm based on Markov chains for finding the minimum cost path in a mobile WSNs," *Int. J. Comput. Commun. Control*, vol. 14, no. 1, pp. 39–55, Feb. 2019.

[8] 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," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 5, pp. 1–10, 2015.

[9] H. B. M. Shashikala, R. George, and K. A. Shujaee, "Outlier detection in network data using thebetweenness centrality," in *Proc. SoutheastCon*, Apr. 2015, pp. 1–5.

[10] B. Li, H. Liu, and W. Su, "Topology optimization techniques for mobile robot path planning," *Appl. Soft Comput.*, vol. 78, pp. 528–544, May 2019.

[11] L. You, E. Yang, and G. Wang, "A novel parallel image encryption algorithm based on hybrid chaotic maps with OpenCL implementation," *Soft Comput.*, vol. 24, no. 16, pp. 12413–12427, Aug. 2020.

[12] Y. Yang, X. Liu, R. H. Deng, and J. Weng, "Flexible wildcard searchable encryption system," *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 464–477, May 2020.

[13] D. Xiaoding Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.

[14] E.-J. Goh, "Secure indexes," *Cryptol. ePrint Arch.*, vol. 2003, p. 216, Dec. 2003.

[15] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf., Comput. Sci. Appl.*, 2008, pp. 1249–1259.

[16] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 79–88.

[17] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2008, pp. 442–455.

[18] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.

[19] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roäu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–16.

[20] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 310–320.

[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. 33rd Annu.Cryptol. Conf.*, 2013, pp. 353–373.

[22] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, "Semantic-aware searching over encrypted data for cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.

[23] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. 16th Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2010, pp. 577–594.

[24] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving Query over encrypted graph-structured data in cloud computing," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 393–402.

[25] Y. Zhang, S. Su, Y. Wang, W. Chen, and F. Yang, "Privacy-assured substructure similarity Query over encrypted graph-structured data in cloud," *Secur. Commun. Netw.*, vol. 13, no. 9, pp. 1933–1944, 2014.

[26] Z. Fan, B. Choi, J. Xu, and S. S. Bhowmick, "Asymmetric structure-preserving subgraph queries for large graphs," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 339–350.

[27] S. Yin, Z. Fan, P. Yi, B. Choi, J. Xu, and S. Zhou, "Privacypreserving reachability Query services," in *Proc. Adv. Appl. Database Syst.*, Bali, IN, USA, 2014, pp. 203–219.

[28] I. Al Ridhawi, Y. Kotb, and Y. Al Ridhawi, "Workflow-net based service composition using mobile edge nodes," *IEEE Access*, vol. 5, pp. 23719–23735, 2017.

[29] Z. Cui, Z. Lu, H. Yang, Y. Zhang, and S. Zhang, "A novel range search scheme based on frequent computing for edge-cloud collaborative computing in CPSS," *IEEE Access*, vol. 8, pp. 80599–80609, 2020.

[30] G. Gao, Z. Cui, and C. Zhou, "Blind reversible authentication based on PEE and CS reconstruction," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 1099–1103, Jul. 2018.

[31] Z. Cui, Q. Qiu, C. Yin, J. Yu, Z. Wu, and A. Deng, "A barrage sentiment analysis scheme based on expression and tone," *IEEE Access*, vol. 7, pp. 180324–180335, 2019.

[32] Z. Wu, R. Wang, Q. Li, X. Lian, G. Xu, E. Chen, and X. Liu, "A location privacy-preserving system based on Query range cover-up or location-based services," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5244–5254, May 2020.

[33] G. Gao, X. Wan, S. Yao, Z. Cui, C. Zhou, and X. Sun, "Reversible data hiding with contrast enhancement and tamper localization for medical images," *Inf. Sci.*, vols. 385–386, pp. 250–265, Apr. 2017.

[34] Z. Xia, L. Jiang, X. Ma, W. Yang, P. Ji, and N. N. Xiong, "A privacy-preserving outsourcing scheme for image local binary pattern in secure industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 629–638, Jan. 2020.

[35] Z. Xia, C. Yuan, R. Lv, X. Sun, N. N. Xiong, and Y.-Q. Shi, "A novel Weber local binary descriptor for fingerprint liveness detection," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 4, pp. 1526–1536, Apr. 2020.

[36] P. Yi, Z. Fan, and S. Yin, "Privacy-preserving reachability Query services for sparse graphs," in *Proc. 30th Int. Conf. Data Eng. Workshops*, Chicago, IL, USA, 2014, pp. 32–35.

[37] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, Apr. 1984.

[38] O. Goldreich, *Foundations of Cryptography*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[39] B. Klimt and Y. Yang, "Introducing the enron corpus," in *Proc. 1st Conf. Email Anti-Spam*, 2004, pp. 1–2.

[40] 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 Math.*, vol. 6, no. 1, pp. 29–123, Jan. 2009.

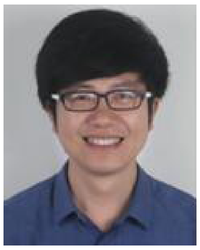
BIN WU received the Ph.D. degree from the Huazhong University of Science and Technology in 2017. He is currently a Lecturer with the School of Information Science and Technology, Jiujiang University. His research interests include privacy preserving, big data security, cloud security, information security, and information query.

XIANYI CHEN received the Ph.D. degree from Hunan University in 2014. He worked as a Visiting Scholar with The University of North Carolina at Pembroke in 2018. He is currently an Associate Professor with the Nanjing University of Information Science and Technology. His research interests include information security of AI, information hiding based on big data, digital forensics, and watermarking in encrypted domain. He is also an Executive Editor of *Journal on Big Data*.

CAICAI ZHANG received the Ph.D. degree from the Huazhong University of Science and Technology in 2016. She is currently an Associate Professor with the School of Information Science and Technology, Jiujiang University. Her research interests include deep learning, data security, and uncertain data management.

ZHUOLIN MEI received the Ph.D. degree from the Huazhong University of Science and Technology in 2017. He is currently an Associate Professor with the School of Information Science and Technology, Jiujiang University. His research interests include privacy protection, cloud computing, information security, and data query.

ZHIQIANG ZHAO received the B.S. degree in computer science and technology from Hunan University, Changsha, China, in 2002, and the Ph.D. degree from the Huazhong University of Science and Technology in 2018. His research interests include visual tracking, computer vision, and image processing.



ZONGDA WU received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), in 2009. From 2012 to 2015, he worked as a Postdoctoral Research Fellow with the University of Science and Technology of China (USTC). He is currently a Professor of computer science with Shaoxing University. His research interests include information retrieval and personal privacy.

TAO YAN received the Ph.D. degree in communication and information systems from Shanghai University, Shanghai, China, in 2010. He has been with the faculty of the School of Information Engineering, Putian University, where he is currently an Associate Professor. His major research interests include multiview high efficiency video coding, rate control, and video codec optimization.