ISSN 2278-3091
Volume 8, No.2, March - April 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse10822019.pdf
https://doi.org/10.30534/ijatcse/2019/10822019

B²EIS-RG: Biometric-based Bucket Encrypting Index Structure with Random Generator

Sudharani K[1], Sakthivel N. K[2] and Subasree S[3]
[1]Research Scholar, Bharathiar University, India, ksudharani.shagthi@gmail.com
[2]Vice Principal, Nehru College of Engineering and Research Centre, India, nksakthivel@gmail.com
[3]Professor and Head, CSE, Nehru College of Engineering and Research Centre, India, drssubasree@gmail.com

ABSTRACT

The proliferation in the popularity of the cloud-based data storage services motivated the data owners to store a huge amount of confidential files on the remote servers in an encrypted format. The users/clients can send their queries to the database owner to retrieve the data files in the encrypted database while protecting privacy of both the queries and the database. The database owners can outsource their enormous biometric data and identification tasks to the cloud server such as Amazon to avoid high storage and computation costs. However, this adds potential threats to the privacy of users. This paper presents a combined Biometric-based Bucket Encrypting Index Structure with Random Generator (B2EIS-RG) for efficient and privacy-preserving biometric identification outsourcing in the cloud. The encryption process includes multi-keyword query processing along with the conjunction and disjunction logic queries to ensure high privacy guarantee against the keyword attacks. Experimental evaluation over a large dataset demonstrates that the proposed scheme can achieve modest time efficiency, and they are practical for use in the huge encrypted database systems. The proposed scheme is found to be highly secure even if the attackers can forge the biometric identification requests.

Key words: Biometric identification, Bucket Encryption, Cloud Computing, Data Outsourcing, Multi-Keyword Search, Privacy Preserving.

1. INTRODUCTION

Database-as-a-Service (DaaS) allows a third party service provider to host the database for the clients, to store and access the cloud databases with the adequate storage resource and low infrastructure cost [1, 2]. The third party provider is in charge of the administrative and maintenance tasks. The database owners can outsource their enormous biometric data and identification tasks to the cloud server such as Amazon to avoid high storage and computation costs. If the user or the data owner opts for more authority over the database, this option is available depending on the third party provider. However, outsourcing the database also raises the data confidentiality and privacy issues due to the loss of data control by the data owner. Encryption of the confidential data before outsourcing is a direct approach to ensure high privacy for the sensitive data [3-8]. Encryption becomes a deterrent to the data utilization capacity while providing strong end-to-end privacy. Also, the users are also concerned about the query privacy, while expecting that the database server should not learn the data or query in the plaintext form.

To address these security and privacy issues, Song et al. [9] introduced Searchable Symmetric Encryption (SSE) for storing data on the mail servers and file servers. A SSE scheme encrypts data in such a way that it can be privately queried through the use of a query-specific token generated with knowledge of the secret key. Kamara et al. [10] proposed a dynamic SSE scheme for real-world cloud storage system. Kamara et al. [11] developed a parallel and dynamic SSE based on the red-black tree data structure. Cash et al. [12] designed dynamic SSE scheme for efficient search of server-held encrypted databases with billions of record-keyword pairs by leveraging the dictionary structure. This scheme shows high scalability while searching on the datasets with billions of document-keyword pairs. Hahn et al. [13] presented a searchable encryption scheme for cloud storage. But, the dynamic SSE schemes do not achieve forward privacy. The search time of these schemes is long with the increase in the number of document-keyword pairs, while updating multiple rounds of communication between the server and client with high overhead.

This work applied a novel index design for processing the queries over the encrypted cloud storage. Based on the structure, two specific structures with privacy guarantee are introduced. These structures make a tradeoff between the data privacy and query efficiency. This design provides data privacy guarantee including forward privacy. The proposed solution has a compact index structure while supporting multi-keyword queries and data updates with moderate overhead. A bucket encrypting index structure with random generator (BEIS-RG) is combined with biometric identification system. Compared with the traditional authentication methods based on the passwords and identification cards, biometric identification is more reliable and convenient [14].

In a biometric identification system, the database owner is responsible to manage the biometric database and outsource the enormous biometric data such as fingerprint, iris, voice patterns, facial patterns, etc., to the cloud server to get rid of the expensive storage and computation costs. However, the biometric data has to be encrypted before outsourcing to

preserve the data privacy [15]. The database owner encrypts the user query and submits it to the cloud. The cloud performs biometric-based identification operations over the encrypted database and returns the result to the database owner. The database owner computes the similarity between the query data and the biometric data associated with the index, and returns the query result to the user. The proposed scheme is found to be highly secure even if the attackers can forge the biometric identification requests.

1.1 Organization of the paper

The paper is systematized in the following order: Section II describes a brief overview of the existing keyword based encryption schemes and privacy preserving schemes. The proposed B^2EIS-RG approach and biometric identification scheme are explained in Section III. The performance analysis of the proposed B^2EIS-RG approach with the existing scheme is presented in Section IV. The concluding statements of the proposed work are discussed in Section V.

2. RELATED WORKS

Fu, et al. [16] devised an effective encryption scheme that supports the multi-keyword ranked search and parallel search capabilities in the cloud. To improve search efficiency, a tree-based index structure is designed to support parallel search, improve the computational capacity and efficiently utilize the cloud server resources. With the parallel search algorithm, the search efficiency is improved. However, the scheme does not focus on the semantics-based search scheme over the encrypted data. Cao, et al. [17] proposed a multi-keyword based ranked search over the encrypted data for better privacy-preserving in the cloud computing environment. The proposed scheme is improved to achieve stringent privacy requirements in different threat models, while requiring minimum computation and communication cost. The integrity of the rank order in the search result is not checked.

Raghavendra, et al. [18] developed a most significant single-keyword based Search algorithm that ensures efficient and secure search in the cloud environment. The indexed keywords are encrypted without incurring overhead from the cloud service provider. Thus, the proposed scheme requires lower computational overhead and time complexity. The proposed algorithm shows significant reduction in the index generation time, index storage space and keyword search time. However, this scheme incurs high search time and index storage space on multimedia. Sun, et al. [19] presented a keyword search scheme based on the attributes with the efficient user revocation for fine-grained authorization in the cloud environment without always depending on the online Trusted Authority (TA). This search scheme allows multiple data owners to independently encrypt the data and outsource the encrypted data to the cloud server. The proposed scheme is selectively secure against chosen-keyword attack. But, the computation cost is higher for this proposed search scheme.

Xia et al. [20] devised a secure multi-keyword ranked search scheme over the encrypted data using Greedy depth-first search algorithm. The secure k-Nearest Neighbor (kNN) algorithm is utilized to encrypt the index and query vectors and ensure accurate relevance score between encrypted index and query vectors. The main drawback of this scheme is dishonest users may distribute their secure keys to the unauthorized ones. Fu et al. [21] proposed an efficient multi-keyword fuzzy ranked search scheme based on the Wang et al.'s scheme. Our proposed scheme enables efficient file update, without requiring a predefined keyword set. A keyword transformation based on the unigram is developed to simultaneously improve the accuracy and create the ability to handle other spelling mistakes. This scheme failed to achieve the ideal state because of the keyword weight.

Dai et al. [22] proposed a verifiable single keyword top-k search scheme that is highly secure against the insider attacks. Data owners generate Verification Codes (VCs) for the corresponding files, which embed the ordered sequence information of the relevance scores between the files and keywords. The interested keyword is returned to the data user together with a VC. The file integrity is verified by the data users through the reconstruction of a new VC on the received files and comparing it with the received one. The proposed scheme does not consider the verifiable multi-keyword based search schemes. Chen et al. [23] developed a verifiable keyword search scheme for the big-data based Mobile healthcare Networks (MHNs) with fine-grained authorization control. In the proposed scheme, while sending the search request to the healthcare provider for the first time, the users need to check whether they possess the right to search within the encrypted data. Only the authorized users can generate valid trapdoors for searching. The main drawback of this scheme is computational overhead.

Peng et al. [24] introduced a tree-based ranked multi-keyword search scheme for multiple data owners. A novel search protocol based on the bilinear pairing is constructed to enable different data owners to use different keys to encrypt their keywords and trapdoors. The proposed scheme needs more storage space for the index. However, this is not a problem for the cloud platform. Ye and Ding [25] proposed a controllable keyword search scheme to address the file security issues. The files that the user has no right to access cannot be retrieved, even if they contain the required keyword. This scheme is useful and practical especially under the circumstances with the hierarchical user groups. Computational cost for decryption is quite high.

Ahmed and Khan [26] formulated a multi-keyword based ranked search over the encrypted data based on the inner product computation and similarity measure of the coordinate matching. It matches with the multiple search results and captures relevance of the data documents in response to the search query. This scheme acquired low overhead on the computation and communication cost. Miao et al. [27] devised a verifiable multi-keyword based search over the encrypted cloud data to protect the data confidentiality and integrity. Through the rigorous security analysis, it is proven that the proposed search scheme is highly secure against the Keyword Guessing Attack (KGA) in the standard model. The empirical experiments over real-world dataset show that the proposed search scheme is efficient and feasible in the practical applications. But, it does not support the expressive search.

Miao et al. [28] created an attribute based multi-keyword search scheme to support comparable attributes by utilizing 0-encoding and 1-encoding. Our proposed scheme can drastically decrease both computational and storage costs. However, the efficiency of the proposed scheme is low. Jiang et al. [29] proposed a multi-keyword ranked search scheme over encrypted cloud data to support search results verification. To reduce the search complexity, the estimated least frequent keyword is initially searched in the query to significantly reduce the number of searching documents. The common rule is applied to calculate the relevance scores of the documents that match with a given search request. This scheme does not verify the rank order of search results.

Wang et al. [30] devised a new ciphertext policy attribute based encryption with the fast keyword search constructions that preserved the fine-grained access control, while supporting hidden policy and fast keyword search. Single keyword search has to be expanded to multi-keyword search without adding additional parameters. Fu et al. [31] designed a novel central keyword scheme with the semantic extension ranked scheme. By extending the central query keyword instead of all keywords, this scheme makes a better tradeoff between the search functionality and efficiency. Our proposed schemes are efficient, effective, and secure. Cao et al. [32] formulated image search schemes based on the Paillier's encryption. A client stores the image data online for convenient data access anywhere and anytime. The encrypted data is uploaded to a cloud server and distance comparison has to be made to represent the similarity scores of these vectors.

Guo et al. [33] proposed a secure multi-keyword ranked search scheme for multiple data owners. A trusted third party is imported to solve the key management issues. But, the computation cost is high. Chen et al. [34] developed a secure multi-keyword ranked search scheme that resists memory leakage attack from inner or external attackers. The proposed scheme utilizes Physically Unclonable Functions (PUFs) to randomize the keywords and document identifiers. Due to the noisy properties of PUFs, the secret keys are recovered using Fuzzy Extractor (FE). To enhance the security of the proposed scheme, an order-preserving function is selected to encode the similarity scores. However, the efficiency of the proposed scheme has to be improved.

Radke et al. [35] applied a hierarchical clustering method to support more search methods and also to complete the demand for fast cipher text search method for big data environments. For the huge number of users and the data in the cloud, it is important for the search method to include multi keyword query. The problem of privacy preserving multi keywords ranked search over the encrypted cloud data can be solved. The main disadvantage is high overhead on computation and communication. Li et al. [36] proposed an Authorized and Ranked Multi-Keyword Search (ARMS) scheme over the encrypted cloud data by leveraging the ciphertext policy attribute-based encryption (CP-ABE) and Symmetric Searchable Encryption (SSE) techniques. The proposed ARMS scheme can achieve confidentiality of documents, trapdoor unlinkability and collusion resistance. The ARMS is more superior and efficient than existing schemes in terms of functionalities and computational overhead. But, the dynamic searchable encryption in cloud computing should be explored.

Many security schemes have been proposed and are mainly focusing on the SSE technique. However, they do not consider the search authorization problem that requires the cloud server only to return the search results to authorized users. Most dynamic SSE solutions leaking information on the updated keywords are vulnerable to the overwhelming file-injection attacks. Dynamic SSE schemes cannot achieve forward privacy. This work presents a combined B^2EIS-RG for efficient and privacy-preserving biometric identification outsourcing in the cloud. The proposed work includes multi-keyword query processing with the conjunction and disjunction logic queries to ensure high privacy guarantee against the keyword attacks.

3. PROPOSED WORK

In this work, three types of entities including the database owner, users and the cloud. The database owner holds a large size of biometric data such as fingerprint that is encrypted and transmitted to the cloud for storage. When a user wants to store the data or access the data in the cloud, a query request is to be sent to the database owner for the identification of the authorized user using the biometric details. After receiving the request, the database owner generates a ciphertext for the biometric trait and then transmits the ciphertext to the cloud for identification. The cloud server figures out the best match for the encrypted query and returns the related index to the database owner. Finally, the database owner computes the similarity between the query data and the biometric data associated with the index, and returns the query result to the user. The FingerCodes are used to represent the fingerprint. A FingerCode consists of 'n' elements and each element is a l-bit integer. If the Euclidean distance between two FingerCodes is below a threshold, they are usually considered as a good match. This means the two fingerprints belongs to the same person. Figure 1 depicts the system model of the proposed B2EIS-RG.

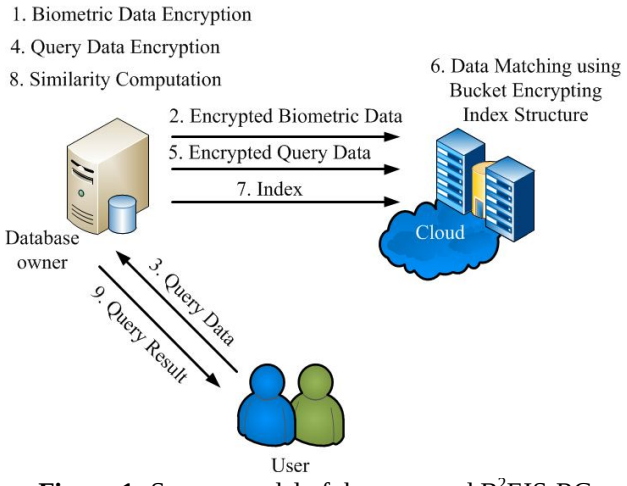


Figure 1: System model of the proposed B^2EIS-RG

In the cloud-based database system, the data owner outsources a huge collection of data files $d = (d_1, \dots, d_{\#d})$ to the remote database server in the encrypted form $c = (c_1, \dots, c_{\#c})$. The data files 'd' can represent the text files or records in a relational database. $w = (w_1, \dots, w_{\#w})$ denotes the keyword universe extracted from the data files. $id(d_j)$ denotes the identifier of the data file d_j and $id(d)$ represents the identifier of all data files. Given a vector 'v', the j[th] element of the vector is referred as $v(j)$ *or* v_j . $div \in \{0,1\}^{\#d}$ denotes a Data Identifier Vector (DIV), where the j[th] element of DIV is 1, if d_j is included. Otherwise, it is zero. div_i or div_{w_i} denote all data files that contain w_i . A multi-keyword query $q = \{w_1, \dots, w_{\#q}\}$ is considered. d_q represents the data files for the conjuction and disjunction logic queries. If $\#q = 1$, then 'w' denotes the single-keyword query, i.e., $q = \{w\}$. d_w denotes all the data files containing the single keyword 'w'. Standard bitwise Boolean operations are defined on the binary vectors such as bitwise OR and bitwise AND operations.

3.1 BEIS-RG

An encoding approach is proposed based on the DIV to identify the files that match with the general multi-keyword queries. Initially, the keywords are mapped onto a line by leveraging a collection of independent hash function and place the corresponding DIVs into the hit buckets. After, both the conjunctive and disjunctive logic queries are achieved by applying arithmetic operations on the DIVs by the issued search token. This is equal to post-processing of a sequence of single keyword queries. The files are returned if the connection of all hit bits in a row equals to one, for the conjunctive logic queries. For the disjunctive logic queries, the files are returned according to the similarity scores computed on all hit bits in a row.

BEIS-I and BEIS-II schemes with privacy preservation are proposed. BEIS-I covers the original bits of the DIVs with the output of a Pseudo Random Function (PRF). A keyed function F is called as a PRF if it is a polynomial-time computable function that there exists no Probabilistic Polynomial-Time (PPT) adversary can distinguish it from random functions when the key is maintained secret. BEIS-II encrypts the bit vectors using Paillier homomorphic cryptosystem. BEIS-I is computationally efficient by using the symmetric encryption. BEIS-II requires minimum communication cost as multiple encrypted DIVs can be combined through the additively homomorphic property by adopting the ciphertext packing technique.

3.1.1 Probabilistic Coding Data Structure

Initialization

Key Generation 1^k: Given a security parameter 'k', a k-bit string sk_1 is sampled uniformly at random, $sk_1 \leftarrow SKE.Gen(1^k)$. $Output\ K = (sk_1, sk_2)$.

$BuildIndex(K, d)$

Step 1: Select a collection of r independent keyed hash functions $h_{i=[1,r]} = \{h_i | i \in [1, r]\}$, where $h_i: \{0,1\}^* \times \{0,1\}^k \rightarrow \{0,1\}^l$.

Step 2: Extract the keyword universe from the data files and generate the corresponding DIVs $\{div_1, \dots, div_{\#w}\}$

Step 3: Initialize an array γ comprising 'm' number of buckets where each bucket is set to be empty.

Step 4: For each $w_i \in w$

Compute the 'r' hash bucket positions $x_1 = h_1(w_i, sk_1), \dots, x_r = h_r(w_i, sk_1)$

For each position x_j, if the bucket $\gamma[x_j]$ is empty, div_i is stored in the corresponding bucket. Otherwise, the bucket is updated by storing the bitwise OR of div_i. The previously stored DIV at the position is denoted by $\gamma[x_j] = \gamma[x_j] \vee div_i$.

Step 5: For $1 \le i \le \#d$, let $c_i \leftarrow SKE.Enc(sk_2, d_i)$.

Step 6: Output (γ, c), where $c = (c_1, \dots, c_{\#c})$

$Recovery(K, c_q)$: $Return\ d_i \leftarrow SKE.Dec(sk_2, c_i)$ for $c_i \in c_q$.

3.1.2 Multi-keyword Search over the index

$\boldsymbol{Trapdoor(sk, q)}$: For each $w_i \in q$, the bucket positions $x_1 = h_1(w_i, sk_1), \dots, x_r = h_r(w_i, sk_1)$ are computed. Output the union of all positions as τ.

$\boldsymbol{Query(\gamma, c, \tau)}$: Use τ to perform logic AND or OR query over the array.

Conjunction logic query: Extract the corresponding DIVs from all hit buckets in the trapdoor τ and compute $div_\tau = \wedge_{y \in \tau} \gamma[y]$. The encrypted data files $c_q = \{c_i \in c: div_\tau[i] = 1$ are returned. To state the correctness, let τw be a sub-trapdoor for a single keyword 'w' in q. Thus, $\tau_w \subset \tau$

$$div_\tau = \wedge_{w \in q} \wedge_{y \in \tau_w} \gamma[y] = \wedge_{w \in q, y \in \tau_w} \gamma[y] = \wedge_{y \in \tau} \gamma[y] \quad (1)$$

The execution will return the data files containing all keywords in the multi-keyword query 'q'.

Disjunction logic query: The similarity score $score(d_j, q)$ is computed between the data file d_j and query 'q'. The score is defined as

$$score(d_j, q) = r \cdot 1(\wedge_{y \in \{\tau[1], \dots, \tau[r]\}} \gamma[y][j] = 1) + \cdots + r \cdot 1(\wedge_{y \in \{\tau[\#qr-r+1], \dots, \tau[\#qr]\}} \gamma[y][j] = 1) \quad (2)$$

Where $1(\cdot)$ denotes the indicator function. The indicator function is equal to 1, if the condition in $1(\cdot)$ holds. Otherwise, it is zero. The similarity scores are sorted in a descending order and the ranked encrypted files are returned.

For the conjunction logic queries, the query result of our basic index structure is equal to post-processing of a sequence of single-keyword queries. For the disjunction logic queries, $score(d_j, q)$ is the number of common buckets mapped by the keywords in the data file d_j and q. Hence, if the number of common buckets is larger, the similarity between d_j and q is high. If a data file contains at least one query keyword, the score is greater than or equal to r. Due to this property, the server can return all the ranked data files with $score(d_j, q) \ge r$ or the ranked results with the top-k highest scores, where k could be user-specified.

3.1.3 Supporting Index Dynamics

Both the addition and removal of the data files should be supported without either re-indexing the whole database from scratch or using the generic and expensive dynamization techniques.

Update Token $(\boldsymbol{K}, \boldsymbol{d_j})$: To add a new data file d_j, the client locally encrypts the data file as c_j. The sub-index γ^* for d_j is constructed as the same structure as γ. The client simply runs $BuildIndex\ (K, d_j)$ and sets $\tau_a = (c_j, \gamma^*)$. To delete an existing data file, the client should initially determine the file location and generate $\tau_d = (j: d_j \in d)$.

Update $(\boldsymbol{\tau_a}/\boldsymbol{\tau_d}, \boldsymbol{\gamma}, \boldsymbol{c})$: For the addition of data file, γ^* is merged into the original index γ and c_j into the ciphertext 'c'. For deletion, set the j[th] entry of each bucket in γ to 0 and delete c_j from the ciphertext.

3.2 BEIS-I: Using Random Generator

The basic index construction supports efficient conjunctive or disjunctive logic queries without requiring privacy preservation. In BEIS-I, every original bit of each IFV is padded with a pseudo number generated by the PRF. By using the symmetric key encryption and inheriting the basic index structure, BEIS-I achieves practical efficiency while performing multi-keyword queries and updates. To avoid reiteration of the same steps, key differences and modifications are only presented.

3.2.1 Initialization and index construction

Key Generation (1^k): In addition, a PRF 'F' defined as $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$ is used. A k-bit string sk_3 is sampled uniformly at a random way to serve as the key of F. Figure 2(a) shows the construction of index structure.

$BuildIndex(K, d)$: After obtaining the basic index structure, additional step to encrypt γ is added.

For each original bit 'b' stored in the index $\gamma[y][j] (y \in 1,m, j \in 1,\#d,$ it is stored as encrypted form

$$\gamma[y][j] = b + \zeta_{yj} \quad (3)$$

Where $\zeta_{yj} = F(sk_1, y \| j \| flag)$. The default value of the flag is set to 0. The value of the flag will be changed to 1, while generating the update token τ_d.

3.2.2 Multi-keyword Search over the encrypted index

$\boldsymbol{Trapdoor(sk, q)}$: The server needs to locate all hit encrypted DIVs for each multi-keyword query and the intermediate encrypted results computed on the specified query logics are returned to the client for further processing.

$\boldsymbol{Query(\gamma, c, \tau)}$: Use τ to compute the intermediate encrypted results according to specified logic.

Conjunction logic query: Locate all the hit encrypted DIVs, compute and return to the client.

$$\sum_{y \in \tau, j \in [1,\#d]} \gamma[y][j] \quad (4)$$

Disjunction logic query: Locate all the hit encrypted DIVs, compute and return to the client.

$$\sum_{y \in \tau[1,r], j \in [1,\#d]} \gamma[y][j], \ldots, \sum_{y \in \tau[\#qr-r+1,\#qr], j \in [1,\#d]} \gamma[y][j] \quad (5)$$

At the client side, ζ_{yj} is recomputed for each $y \in \tau$ and $j \in [1, \#d]$ by leveraging the secret key sk_1. According to different query logics

Conjunction logic query: The client locally computes $st_q = (st_{q,1}, \ldots, st_{q,\#d})$ where

$$st_{q,j} = \sum_{y \in \tau} (1 + \zeta_{yj}) \ (j \in [1, \#d]) \quad (6)$$

Then, an empty set ID is initialized and corresponding identifier $id(d_j) (j \in [1, \#d])$ is added into ID, if

$$\sum_{y \in \tau} \gamma[y][j] = st_{q,j} \quad (7)$$

Disjunction logic query: The client locally computes $st_q = (st_{q,1}, \ldots, st_{q,\#d})$ where

$$st_{q,j} = \left(\sum_{y \in \tau[1,r]} \zeta_{yj} + r, \ldots, \sum_{y \in \tau[\#qr-r+1,\#qr]} \zeta_{yj} + r\right) \quad (8)$$

For each $j \in [1, \#d]$, the similarity score is computed as follows

$$score(d_j, q) = r \cdot 1\left(\sum_{y \in \tau[1,r]} \gamma[y][j] = st_{q,j}[1]\right) + \cdots + r \cdot 1\left(\sum_{y \in \tau[\#qr-r+1,\#qr]} \gamma[y][j] = st_{q,j}[\#q]\right) \quad (9)$$

Then, the scores are sorted in a descending order and corresponding identifier $id(d_j) (j \in [1, \#d])$ is added into the ID, if $score(d_j, q)$ is greater than a predefined threshold. For both the cases, the client sends ID to the server. Finally, the server returns the encrypted data files c_q to the client.

3.2.3 Supporting Index Dynamics

The BEIS-I scheme can support efficient data updates. Following updates are performed based on the basic index construction.

Update Token $(\boldsymbol{K}, \boldsymbol{d_j})$: To add a new data file d_j, the client locally encrypts each bit of the newly obtained sub-index γ^* in a similar way processed in BuildIndex and outputs encrypted sub-index and c_j as τ_a. The client generates an m-dimensional row vector "Del" to delete an existing data file, where each entry is initialized to zero. Then, the encrypted version is generated by computing $Del_j[i] = 0 + \zeta_{ij}$, where $i \in [1, m]$ and $\zeta_{ij} = F(sk_1, i \| j \| flag + 1)$. Finally, the encrypted row vector is obtained as output as τ_d.

Update $(\boldsymbol{\tau_a}/\boldsymbol{\tau_d}, \boldsymbol{\gamma}, \boldsymbol{c})$: For the addition of data file, the encrypted sub-index γ^* extracted from τ_a is added to γ and c_j is added into c. For the deletion of file, the j[th] row of γ is set to the encrypted row vector "Del" extracted from τ_d and c_j is deleted from the ciphertext accordingly.

3.3 BEIS-II: Using Homomorphic Generator

BEIS-II is introduced by leveraging the Paillier cryptosystem to reduce the communication cost and encrypt the DIVs stored in each bucket. Due to the additive homomorphic property, multiple DIVs can be aggregated directly at the server side in an encrypted form.

3.4 Ciphertext packing

The typical parameters for the Paillier cryptosystem can support big integers as the plaintext space. The bit vectors of length #d are to be encrypted and each bit vector is to be packed into multiple 1024-bit integers. To ensure the addition of bit values in one row across 'r' different buckets will not affect the aggregated values of other rows during the query phase. There is a need to allocate $[\log_2(r+1)]$ bits for each original bit, $([\log_2(r+1)]-1)$ bit of zeros before each bit. Our deletion mechanism requires the usage of an integer to indicate the deletion of a certain file, more extra zeros need to be inserted before each bit. For the removal of existing data file, the second to the last entry is set to 1 in 1024-bit vector corresponding to the data file and its encrypted version is generated. After, the encrypted vector is used to perform additive homomorphic operations on all the DIVs stored in the buckets.

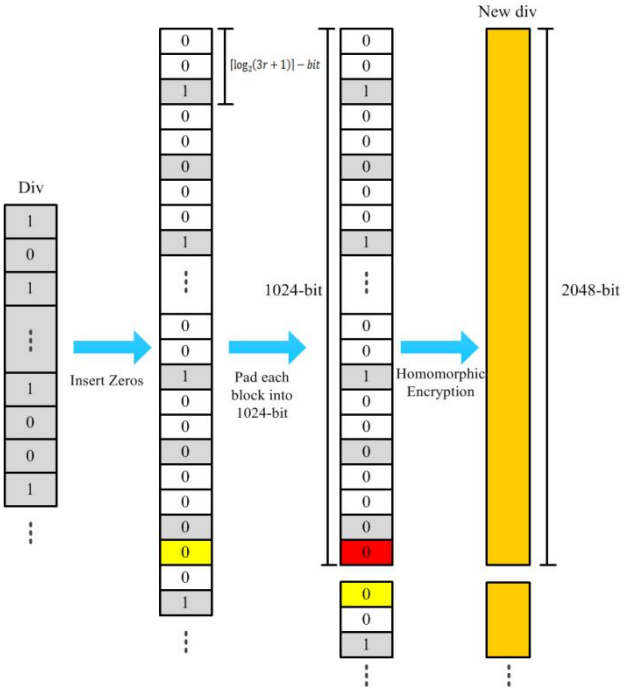


Figure 2(a): Example of Index Building

Upon finishing the updates, last two entries corresponding to the data files in all the buckets are modified to either "10" or "11". For any new future query, the decimal value of entries corresponding to the data files in the aggregated form is turned to 3r or 2r. This does not incur collision and affect the precision of checking equality for a new query. A block of $[\log_2(3r+1)]$ bits is used in the plaintext space to represent a bit in the DIV and $([\log_2(3r+1)]-1)$ bit of zeros is padded before each original bit. Moreover, the remaining bits that are inadequate to form a block are set to zero. At last, the packed integers are encrypted by the Paillier cryptosystem.

3.4.1 Initialization and Index construction

Key Generation (1^k): Generate (sk_p, pk_p) for the Paillier cryptosystem and the new key tuple is added to K.
$BuildIndex(K, d)$: Based on the basic index structure, the additional steps are performed as follows

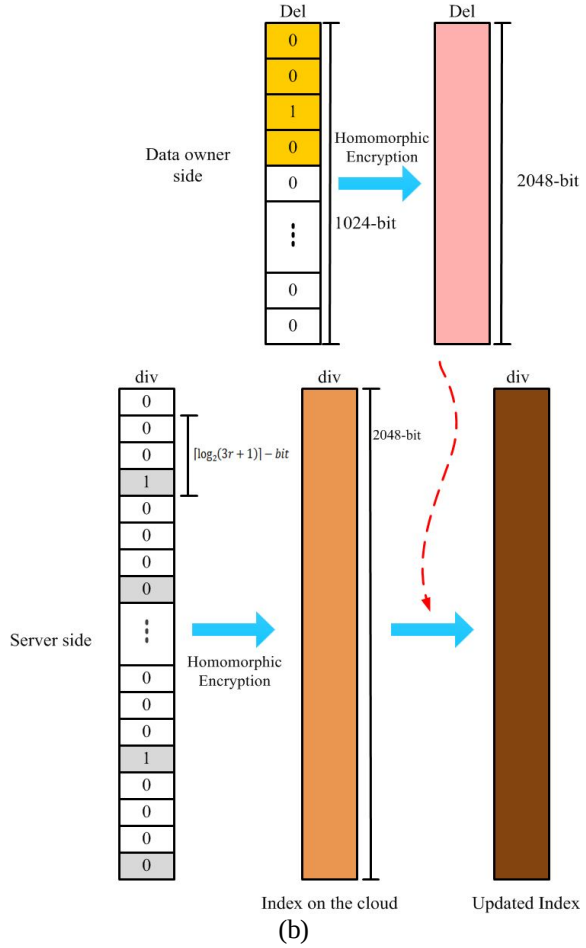


Figure 2(b): Example of deleting a file

Extract div stored in $\gamma[y](y \in [1,m])$ and construct $\widehat{div}$ by inserting the padded bits before each bit of div. $\widehat{div}$ is packed into multiple 1024-bit plaintext blocks and Paillier cryptosystem is applied to encrypt each block under the public key pk_p . The newly obtained $div = [\![block_{y,1}]\!] \| \ldots \| [\![block_{y,v}]\!]$ is stored into $\gamma[y]$, where v represents the maximum number of blocks.

3.4.2 Multi-keyword search over the Encrypted index

$\boldsymbol{Trapdoor\ (sk, q)}$: Similar to the basic index construction. After locating the hit buckets, the server homomorphically aggregates these encrypted DIVs and returns the intermediate aggregation result to the client.
$\boldsymbol{Query(\gamma, c, \tau)}$: Use τ to compute the intermediate encrypted results as follows. For $t \in [1, \#q]$, compute

$\theta_t = \prod_{y\in\tau[1,r]}[\![block_{y,1}]\!] \| \dots \| \prod_{y\in\tau[\#qr-r+1,\#qr]}[\![block_{y,v}]\!]$ (10)

And return $\Theta = \{\theta_1, \dots, \theta_{\#q}\}$ to the client.

After the client decrypts the intermediate ciphertext Θ using sk_p, reveals the blocks and analyzes the decryption result as a concatenation of decimal values based on the aligned configuration of the packing technique. The decrypted binary string θ_t is converted to $B_t = (b_1 \dots b_{\#d})_{10} (t \in [1, \#q])$, where b_i denotes the binary representation of $[\log_2(3r+1)]$ bit integer.

Conjunction logic query: Add the corresponding $id(d_j)$ into ID if $\forall t \in [1, \#q], B_t[j] = r$.

Disjunction logic query: For each $j \in [1, \#d]$, the similarity score is computed.

$score(d_j, q) = r \cdot 1(B_1[j] = r) + \dots + r \cdot 1(B_{\#q}[j] = r)$ (11)

The scores are sorted in a descending order and corresponding $id(d_j)$ is added into ID if the score is greater than a predefined threshold.

3.4.3 Supporting Index Dynamics

Finally, the file updates are shown by the BEIS-II scheme.

$\boldsymbol{UpdateToken}(\boldsymbol{K}, \boldsymbol{d_j})$: To add a new file d_j, for the sub-index γ^* generated in the basic index construction, the client initially pads the sub-index to the binary strings of 1024 bits using the same packing method. Then, the sub-index is encrypted using Paillier cryptosystem under the pk_p. The encrypted sub-index and ciphertext are output as τ_a. To delete an existing data file, the client initializes a 1024-bit zero vector Del, $\left(\left(j \bmod \left\lfloor \frac{1024}{\lceil \log_2(3r+1) \rceil} \right\rfloor\right) \cdot \lceil \log_2(3r+1) \rceil - 1\right)$ th entry is set to 1 and encrypted by leveraging the Paillier homomorphic encryption. The encrypted Del is output as τ_d.

Update $(\boldsymbol{\tau_a}/\boldsymbol{\tau_d}, \boldsymbol{\gamma}, \boldsymbol{c})$: Procedures for adding a new data file are similar for the basic index construction. For file deletion, for the $\frac{j \cdot \lceil \log_2(3r+1) \rceil}{1024}$ th ciphertext block in all the buckets $\gamma[y] (y \in [1, m])$ and encrypted Del extracted from τ_d. The server computes $\gamma[y] \leftarrow \gamma[y] \cdot \mathrm{Del}$ by restoring to the additively homomorphic property and deletes c_j from c. This works, as the decimal value of entries corresponding to the data file is changed from r to 3r.

Figure 2(b) shows the file deletion process. If the client wants to delete the file d_1, this is implied in the first ciphertext block of each div. The client initializes a 1024-bit vector $Del = (0010, 00 \dots)$ and encrypts the vector with the homomorphic encryption. After receiving the Del, the server multiplies it with the first block of each div to complete the update. Due to the fact that the message space of the Paillier cryptosystem is always greater than the space allocated for a data file. Therefore, the updates can be achieved in a batch and collection of data files can be added and removed simultaneously [37].

3.5 Biometric identification scheme

The ciphertext is reconstructed to reduce the amount of uploaded data and the efficiency in the preparation and identification procedures are improved.

3.5.1 Preparation

In the preparation process, b_i is the i[th] sample feature vector derived from the fingerprint image using a feature extraction algorithm [38]. To be more specific, b_i is an n-dimensional vector with 'l' bits of each element.

For easy identification, the fingerprint code b_i is extended by adding $(n+1)$ th element as B_i. Then, the database owner encrypts B_i with the secret key sk as follows

$C_i = B_i \times sk$ (11)

The database owner performs the following operation

$C_h = M_2^{-1} \times H^T$ (12)

Each Unique ID B_i is associated with an index I_i. After executing the encryption operations, the database owner uploads (C_i, C_h, I_i) to the cloud.

3.5.2 Identification

The steps in the identification process are described below

Step 1: When the fingerprint of a user to be identified, the query Unique ID b_c derived from the query fingerprint image. The Unique ID b_c is a n-dimensional vector. Then, the user sends the Unique ID to the database owner.

Step 2: After receiving Unique ID, the database owner extends b_c to B_c by adding $(n+1)$th element equals to 1. Then, the database owner randomly generates a $(n+1) \times (n+1)$ matrix 'E'. The i[th] row vector $E_i = [E_{i1}, E_{i2}, \dots, E_{i(n+1)}]$ is set as a random vector, where the $(n+1)$ th element is $(1 - \sum_{j=1}^{n} E_{ij} * H_j)/H_{n+1}, where\ 1 \le i \le (n+1)$. Then, the database owner performs the following computation to hide Unique ID

$F_c = [E_1^T * b_{c1}, E_2^T * b_{c2}, \dots, E_{(n+1)}^T * b_{c(n+1)}]^T$ (13)

To send F_c to the cloud, the database owner needs to encrypt F_c with the secret keys and a random integer r, $r > 0$. The computation is performed as follows

$C_f = M_1^{-1} \times r \times F_c \times M_2$ (14)

Then, the database owner sends the C_f to the cloud for identification purpose.

Step 3: After receiving C_f from the database owner, the cloud starts to search the Unique ID having minimum Euclidean distance with respect to the query Unique ID. P_i represents the relative distance between B_i and B_c as follows

$P_i = C_i \times C_f \times C_h$

$= B_i \times M_1 \times M_1^{-1} \times r \times F_c \times M_2 \times M_2^{-1} \times H^T$

$= B_i \times r \times F_c \times H^T$

$P_i = \sum_{j=1}^{n+1} r * b_{ij} * b_{cj}$ (15)

In the above equation, the computation result can be used to compare two Unique IDs. To compare the query b_c with two Unique IDs b_i and b_z, the cloud computes P_i and P_z and performs the following operation, where $1 \le i, z \le t, i \ne z$

$$P_i - P_z = \sum_{j=1}^{n+1} r * b_{ij} * b_{cj} - \sum_{j=1}^{n+1} r * b_{zj} * b_{cj} \quad (16)$$
$$= 0.5r(dist_{zc}^2 - dist_{ic}^2)$$

As shown in the above equation, if $P_i - P_z > 0$, the cloud learns that b_i matches with the query unique code much better than b_z. After repeating the operations for the encrypted unique code database 'C' in the cloud, the ciphertext C_i having minimum Euclidean distance with b_c is found out. The cloud obtains the corresponding index I_i according to the tuple (C_i, C_h, I_i) and sends it back to the database owner.

Step 4: After receiving the index, the database owner obtains the corresponding sample unique code b_i in the database 'D' and computes the accurate Euclidean distance between b_i and b_c as $dist_{ic} = \sqrt{\sum_{j=1}^{n}(b_{ij} - b_{cj})^2}$. Then, the database owner compares the Euclidean distance with the standard threshold. The query is identified, if the Euclidean distance is lesser than the threshold value. Otherwise, the identification fails.

Step 5: Finally, the database owner returns the identification result to the user [15].

4. PERFORMANCE ANALYSIS

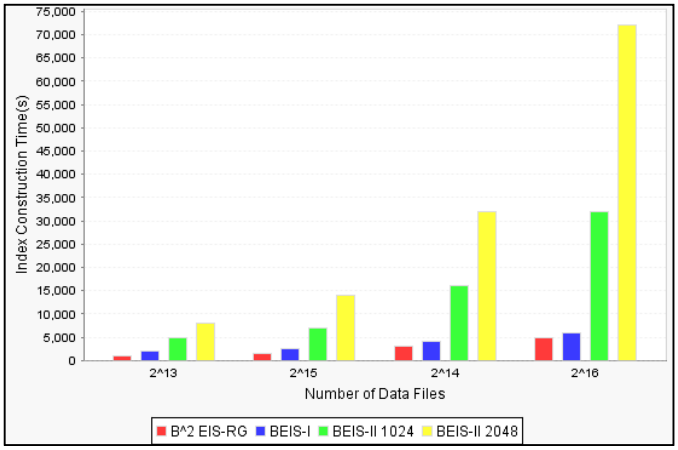
The proposed work B^{2}EIS-RG is evaluated and compared with the BEIS-I [37] and Zhu et al. scheme [15]. The BEIS schemes can execute multi-keyword Boolean search while achieving forward privacy and a strong privacy guarantee that the server cannot learn whether a newly added file contains a previously searched keyword or not. In addition, the search tokens/keyword hashes of all keywords in the new file are not leaked. If the forward privacy is not required, more efficient file additions are obtained. BEIS schemes can naturally support multi-keyword search, without needing to post-process all results of single keyword queries.

Figure 3 shows the time cost analysis of the index construction process. During the index construction of BEIS-I, an 80-bit random number is used to mask each entry in div. This leads to only a small computation cost. While during the index construction of BEIS-II, the use of Paillier homomorphic encryption system to encrypt each div leads to relatively higher computation cost. The B^{2}EIS-RG scheme requires minimum computational time for the index construction when compared to the BEIS-I and BEIS-II schemes.

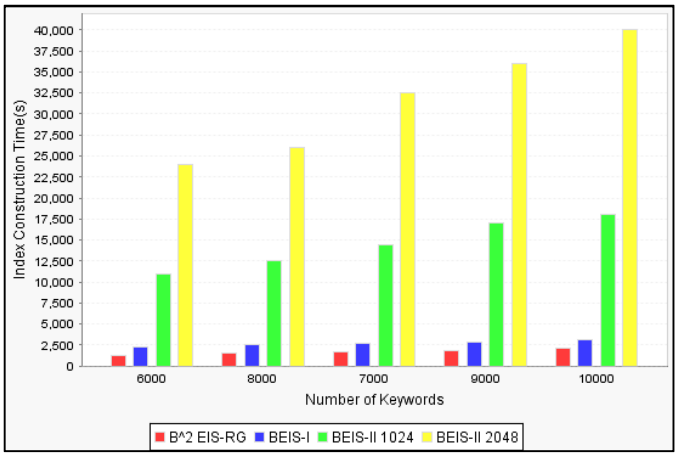
Figure 4 shows the query performance analysis. The time cost of query process consists of the time cost of generating trapdoors on the client side and the time cost of file searching on the server side. Our BEIS schemes can naturally support multi-keyword search, without needing to post-process all results of single-keyword queries. The performance of the single keyword query and multi keyword query is evaluated with respect to the increase in #d. The computation cost for the conjunction and disjunction logics is same.

Figure 5 depicts the time cost of the addition and deletion operations. The cost of file updates for the BEIS-I scheme increase with the increase in the number of data files to be updated. Hence, both the addition and deletion operations are efficient. Due to the unique encryptions of the index in the proposed scheme, the time costs of file updates remain invariant. The proposed scheme can be better applied to the updates of a bunch of data files simultaneously. During the process of file addition, the time costs of the proposed scheme with and without achieving forward privacy are measured. For without forward privacy, there is no need to generate m encrypted value for every bucket. To insert each data file, the client can only compute a single encrypted value of a 1024-bit vector and send the 2048-bit ciphertext to the server along with the position information of the buckets to be changed. This can greatly reduce the time costs for generating adding token if forward privacy is not required in practice.

Figure 6 and Figure 7 present the computation and communication costs in the identification phase with the number of FingerCodes ranges from 1000 to 5000. The computation and communication costs will increase linearly with the increase in the size of the database. Due to the fewer vector matrix multiplication, the identification time and cost are reduced significantly when compared with the Yuan and Yu scheme [39] and Wang et al. scheme [40]. The bandwidth cost of all schemes is almost same, due to the transmission of matrix in the identification phase.

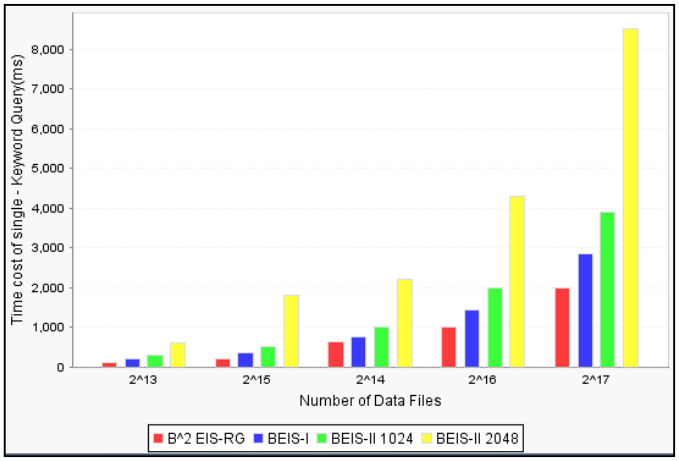


(a)

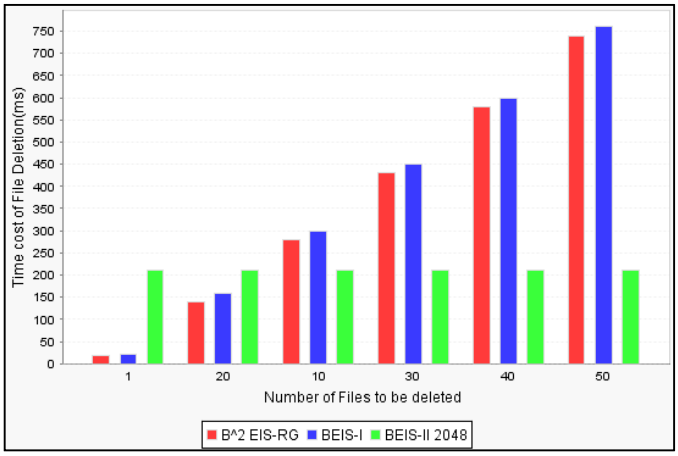


(b)

Figure 3: Average time cost of index construction

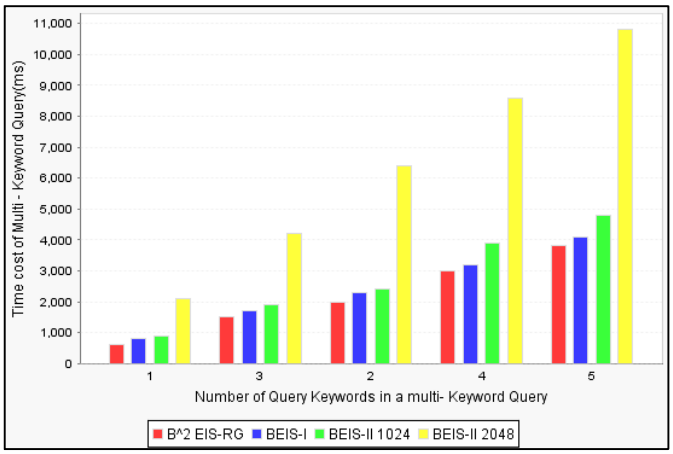


(a)



(b)

Figure 5: Average time cost of file updates



(b)

Figure 4: Average time cost of query process

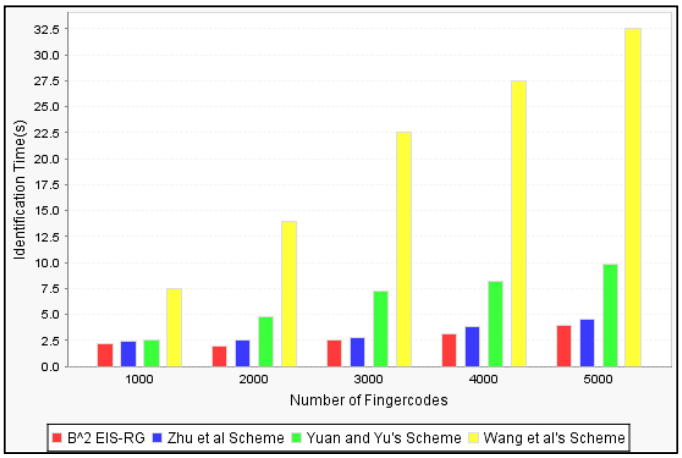
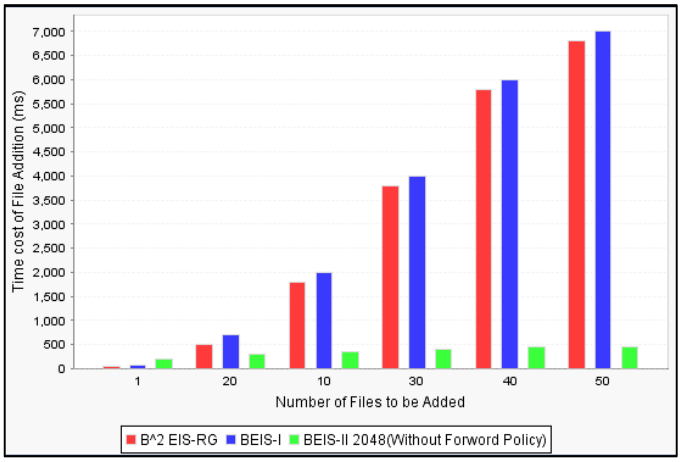


Figure 6: Time cost in the identification phase



(a)

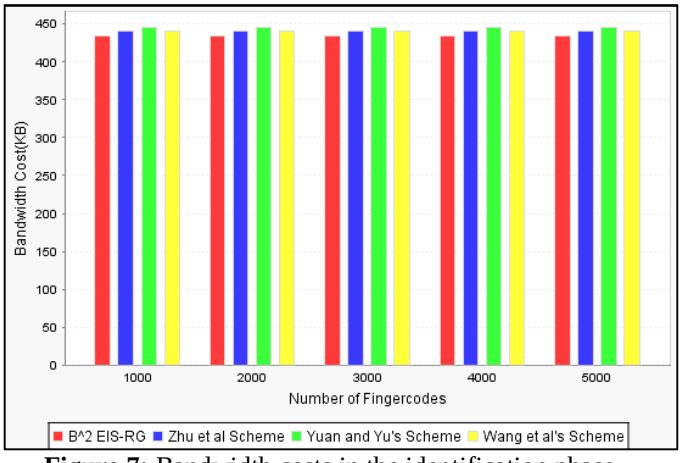


Figure 7: Bandwidth costs in the identification phase

5. CONCLUSION

This paper presents a combined biometric identification scheme and bucket encrypting index structure with random generator for efficient privacy preserving in the cloud environment. The proposed scheme can resist the potential attacks. A new secure index design is applied for processing queries over large-scale encrypted databases. Our index constructions addressed the trade-offs between the query efficiency and query privacy, with flexible and complete query functionalities. Through the extensive, experiments on real-world datasets, the effectiveness and practicality of our constructions are demonstrated. From the experimental analysis, it is observed that the proposed scheme incurs minimum average time for index construction process, query process, file update process, minimum identification time and bandwidth cost than the existing schemes.

REFERENCES

[1] Dr. Abdelrahman ElSharif Karrar and and M. F. I. Fadl, "Security Protocol for Data Transmission in Cloud Computing," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 7, pp. 1-5, Jan-Feb 2018. https://doi.org/10.30534/ijatcse/2018/01712018

[2] Goodubaigari Amrulla, Murlidher Mourya, R. R. Sanikommu, and a. A. A. Afroz, "A Survey of : Securing Cloud Data under Key Exposure," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 7, pp. 30-33, Jan-Feb 2018. https://doi.org/10.30534/ijatcse/2018/01732018

[3] K. He, Jing Chen, Ruiying Du, Qianhong Wu, a. Guoliang Xue, and X. Zhang, "Deypos: Deduplicatable dynamic proof of storage for multi-user environments," *IEEE Transactions on Computers*, vol. 65, pp. 3631-3645, Dec 2016. https://doi.org/10.1109/TC.2016.2560812

[4] S. Hu, Qian Wang, Jingjun Wang, Zhan Qin, and a. K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Transactions on Image Processing*, vol. 25, pp. 3411-3425, May 2016. https://doi.org/10.1109/TIP.2016.2568460

[5] J. Shen, Jun Shen, Xiaofeng Chen, Xinyi Huang, and a. W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 2402-2415, Oct 2017. https://doi.org/10.1109/TIFS.2017.2705620

[6] K. M. R. Urs, "Harnessing the Cloud for Securely Outsourcing Large-scale Systems of Linear Equations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1172-1181, Jun 2013. https://doi.org/10.1109/TPDS.2012.206

[7] Q. Wang, Shengshan Hu, Kui Ren, Meiqi He, Minxin Du, and a. Z. Wang, "Cloudbi: Practical privacy-preserving outsourcing of biometric identification in the cloud," *In European Symposium on Research in Computer Security, Springer, Cham*, pp. 186-205, Sep 2015. https://doi.org/10.1007/978-3-319-24177-7_10

[8] Q. Wang, Cong Wang, Kui Ren, Wenjing Lou, and a. J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE transactions on parallel and distributed systems*, vol. 22, pp. 847-859, May 2011. https://doi.org/10.1109/TPDS.2010.183

[9] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000, pp. 44-55.

[10] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965-976. https://doi.org/10.1145/2382196.2382298

[11] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *International Conference on Financial Cryptography and Data Security*, 2013, pp. 258-274. https://doi.org/10.1007/978-3-642-39884-1_22

[12] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, *et al.*, "Dynamic searchable encryption in very-large databases: data structures and implementation," in *NDSS*, vol. 14, pp. 23-26, Feb 2014. https://doi.org/10.14722/ndss.2014.23264

[13] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 310-320. https://doi.org/10.1145/2660267.2660297

[14] A. Jain, L. Hong, and S. Pankanti, "Biometric identification," *Communications of the ACM*, vol. 43, pp. 90-98, Feb 2000. https://doi.org/10.1145/328236.328110

[15] L. Zhu, C. Zhang, C. Xu, X. Liu, and C. Huang, "An Efficient and Privacy-Preserving Biometric Identification Scheme in Cloud Computing," *IEEE Access*, vol. 6, pp. 19025-19033, 2018. https://doi.org/10.1109/ACCESS.2018.2819166

[16] 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, pp. 190-200, Jan 2015. https://doi.org/10.1587/transcom.E98.B.190

[17] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, pp. 222-233, Jan 2014. https://doi.org/10.1109/TPDS.2013.45

[18] S. Raghavendra, C. Geeta, K. Shaila, R. Buyya, K. Venugopal, S. Iyengar, *et al.*, "MSSS: most significant single-keyword search over encrypted cloud data," in *Proceedings of the 6th Annual*

Intrernational Conference on ICT: BigData, Cloud and Security, 2015. https://doi.org/10.5176/2382-5669_ICT-BDCS15.22

[19] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 1187-1198, Apr 2016. https://doi.org/10.1109/TPDS.2014.2355202

[20] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, pp. 340-352, Feb 2016. https://doi.org/10.1109/TPDS.2015.2401003

[21] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, pp. 2706-2716, Dec 2016. https://doi.org/10.1109/TIFS.2016.2596138

[22] H. Dai, X. Zhu, G. Yang, and X. Yi, "A Verifiable Single Keyword Top-k Search Scheme against Insider Attacks over Cloud Data," in *3rd International Conference on Big Data Computing and Communications (BIGCOM)*, 2017, pp. 111-116. https://doi.org/10.1109/BIGCOM.2017.56

[23] Z. Chen, F. Zhang, P. Zhang, J. K. Liu, J. Huang, H. Zhao, *et al.*, "Verifiable keyword search for secure big data-based mobile healthcare networks with fine-grained authorization control," *Future Generation Computer Systems*, vol. 87, pp. 712-724, Oct 2018. https://doi.org/10.1016/j.future.2017.10.022

[24] T. Peng, Y. Lin, X. Yao, and W. Zhang, "An Efficient Ranked Multi-Keyword Search for Multiple Data Owners Over Encrypted Cloud Data," *IEEE Access*, vol. 6, pp. 21924-21933, 2018. https://doi.org/10.1109/ACCESS.2018.2828404

[25] J. Ye and Y. Ding, "Controllable keyword search scheme supporting multiple users," *Future Generation Computer Systems*, vol. 81, pp. 433-442, Apr 2018. https://doi.org/10.1016/j.future.2017.09.030

[26] M. Ahmed and A. Khan, "Privacy Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," 2018.

[27] Y. Miao, J. Ma, X. Liu, Z. Liu, L. Shen, and F. Wei, "VMKDO: Verifiable multi-keyword search over encrypted cloud data for dynamic data-owner," *Peer-to-Peer Networking and Applications*, vol. 11, pp. 287-297, Mar 2018. https://doi.org/10.1007/s12083-016-0487-7

[28] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, Vol. 5, pp. 3008-3018, Aug 2017. https://doi.org/10.1109/JIOT.2017.2779124

[29] X. Jiang, J. Yu, J. Yan, and R. Hao, "Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data," *Information Sciences*, vol. 403, pp. 22-41, Sep 2017. https://doi.org/10.1016/j.ins.2017.03.037

[30] H. Wang, X. Dong, and Z. Cao, "Multi-value-Independent Ciphertext-Policy Attribute Based Encryption with Fast Keyword Search," *IEEE Transactions on Services Computing*, Sep 2017. https://doi.org/10.1109/TSC.2017.2753231

[31] Z. Fu, X. Wu, Q. Wang, and K. Ren, "Enabling central keyword-based semantic extension search over encrypted outsourced data," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 2986-2997, Dec 2017. https://doi.org/10.1109/TIFS.2017.2730365

[32] Z. Cao, C. Mao, L. Liu, W. Kong, and J. Wang, "Analysis of One Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," *IJ Network Security*, vol. 20, pp. 683-688, Jul 2018.

[33] Z. Guo, H. Zhang, C. Sun, Q. Wen, and W. Li, "Secure multi-keyword ranked search over encrypted cloud data for multiple data owners," *Journal of Systems and Software*, vol. 137, pp. 380-395, Mar 2018. https://doi.org/10.1016/j.jss.2017.12.008

[34] L. Chen, L. Qiu, K.-C. Li, and S. Zhou, "A secure multi-keyword ranked search over encrypted cloud data against memory leakage attack," *Journal of Internet Technology*, vol. 19, pp. 167-176, Jan 2018.

[35] D. Radke, N. Hatwar, L. Gouda, M. Shambharkar, P. Gajimwar, and R. Raut, "An Efficient Search Method over an Encrypted Cloud Data," 2018.

[36] H. Li, D. Liu, K. Jia, and X. Lin, "Achieving authorized and ranked multi-keyword search over encrypted cloud data," in *IEEE International Conference on Communications (ICC)*, 2015, pp. 7450-7455. https://doi.org/10.1109/ICC.2015.7249517

[37] M. Du, Q. Wang, M. He, and J. Weng, "Privacy-Preserving Indexing and Query Processing for Secure Dynamic Cloud Storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2320-2332, Sep 2018. https://doi.org/10.1109/TIFS.2018.2818651

[38] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *IEEE transactions on Image Processing*, vol. 9, pp. 846-859, May 2000. https://doi.org/10.1109/83.841531

[39] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 2652-2660. https://doi.org/10.1109/INFCOM.2013.6567073

[40] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang, "Cloudbi: Practical privacy-preserving outsourcing of biometric identification in the cloud," in *European Symposium on Research in Computer Security*, 2015, pp. 186-205. https://doi.org/10.1007/978-3-319-24177-7_10