Secure and Efficient Bloom Filter-based Image Search in Cloud-based Internet of Things

Yingying Li, Jianfeng Ma, Yinbin Miao, Xiangyu Wang, Rongxing Lu, *Fellow, IEEE,* and Wei Zhang

***Abstract*—Image search is a hot topic, which has played a significant role in various Internet of Things (IoT) applications, such as disease diagnosis, face recognition, and fingerprint recognition. Meanwhile, the proliferation of images has led image owners to outsource images to the cloud for reducing local storage and computation burdens. Therefore, image search without compromising privacy over cloud has received considerable attention and extensively explored in the literature. Many Bloom filter-based schemes have been put forth in past years, however most of them suffer from high storage overhead, low false positive rate, and even expose the values in Bloom filter. To solve these challenges, in this paper, we first design a Merged and Repeated Indistinguishable Bloom Filter (MRIBF) index structure, which can reduce the storage overhead and achieve adaptive security with a low false positive rate. Then, with the MRIBF, we propose a secure and efficient Bloom Filter-based Image Search scheme (BFIS) to achieve a faster-than-linear and more accurate search. Detailed theoretical analysis shows that our scheme is really accurate and secure. Extensive experiments demonstrate that our scheme is indeed efficient and feasible.**

***Index Terms*—Image search, Bloom filter, clustering.**

I. Introduction

CONTENT-BASED image search, which aims to identify images that are similar to the interesting image, has a significant number of applications in various areas, such as disease diagnosis, face recognition, and fingerprint recognition. Since tremendous volumes of data with high velocity are produced in the Internet of Things (IoT) [1], data owners with limited computational and storage resources prefer to outsource their data to a powerful cloud and delegate the cloud server to provide search services for users as shown in Fig. 1. However, the data is sensitive and the cloud server is not fully trusted, outsourcing plaintext data directly to the cloud will lead to privacy leakage. Encryption is a viable approach, but it will hinder the cloud server from performing searches over outsourced ciphertexts.

This work was supported by the National Key Research and Development Program of China (No. 2021YFB3101100), the National Natural Science Foundation of China (No. 62072361, No. 62202364, No. 62202377), the Fundamental Research Funds for the Central Universities (No. ZYTS23166), the Key Research and Development Program of Shaanxi Province (No. 2022GY-019), the Natural Science Basic Research Plan in Shaanxi Province of China (No. 2022JM-353), the Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 22JK0560).

Y. Li is with the School of Mathematics, Hangzhou Normal University, Hangzhou 311121, China (e-mail: lyylyingying@163.com).

J. Ma, Y. Miao, X. Wang, and W. Zhang are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China; Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China (e-mail: jfma@mail.xidian.edu.cn; ybmiao@xidian.edu.cn; xywang_xidian@163.com; ADGJ17861428241@163.com).

R. Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

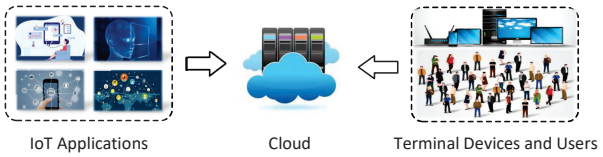

Fig. 1. A cloud-assisted search example.

Homomorphic Encryption (HE), which supports ciphertext computation, can achieve image search over ciphertext. However, encrypted image search schemes [2]–[5] that encrypt feature vectors with HE algorithms (*e.g.,* Paillier encryption) are not suitable for IoT scenarios due to high computational and storage overheads. Although the Asymmetric Scalar-product Preserving Encryption (ASPE) algorithm [6] can greatly reduce the overhead and achieve efficient search over ciphertext without decryption [7]–[11], it has been proven to be actually insecure against even Ciphertext-Only Attack (COA) [12]. Moreover, both homomorphic encryption and ASPE need to encrypt each feature vector of each image and then calculate their Euclidean distances, which results in an increase in overhead as the total number of images increases. The increased overhead will be unbearable when deployed in large-scale datasets.

To reduce the overhead, it is an alternative to represent the image feature vectors with keywords and implement an approximate search based on Bloom Filter (BF) [13], which has been extensively explored in academic and industrial fields. The traditional BF-based schemes build a BF for each data, resulting in a waste of storage space [14], [15]. Moreover, to hide the values in BF, encrypting BF with the Hidden Vector Encryption (HVE) algorithm [16] further increases the storage overhead of BF [17]. Twin Bloom Filter (TBF) [18], [19] masks the values in BF with a hash function and a random number instead of complex cryptographic algorithms, which can protect the values in BF and achieve adaptive security. However, the storage overhead of TBF is still twice that of BF. Circular Shift and Coalesce Bloom Filter (CSCBF) [20] effectively utilizes the storage space and accelerates the search process by merging and repeating BF, but it will expose the values in BF.

Inspired by TBF, we modify the CSCBF to build a novel index, namely merged and repeated indistinguishable Bloom filter (MRIBF). MRIBF masks the values in Bloom filters to achieve adaptive security and repeats multiple Bloom filters to reduce the false positive rate. With the MRIBF, a secure and efficient Bloom Filter-based Image Search scheme (BFIS) is

proposed. Specifically, the main contributions of our work are three-fold as follows.

- First, we design a novel index structure MRIBF based on Bloom filter, which not only reduces storage overhead and false positives rate but also achieves adaptive security.
- Second, we propose a secure and efficient image search scheme based on MRIBF, namely BFIS, which improves security and search efficiency.
- Third, we analyze the security of BFIS and conduct extensive experiments to evaluate its performance. The results show that our BFIS is adaptively secure and at least five times faster than the existing schemes.

The remainder of this paper is organized as follows. In Section II, we introduce some related work. Then, we describe some preliminaries in Section III and introduce our system model, threat model, problem definition, and design goals in Section IV. Later, we present our scheme in Section V, followed by theoretical analysis and performance evaluation in Sections VI and VII, respectively. Finally, we conclude our work in Section VIII.

II. Related work

In this section, we briefly review some recently proposed encrypted image search schemes.

Zhang *et al.* [2] protected the image features with the Paillier homomorphic encryption algorithm for performing search over encrypted images. Later, with the multi-level homomorphic encryption algorithm [21], Zhang *et al.* [3] presented a privacy-preserving image search scheme supporting multi-key multi-user settings. To improve search efficiency, Li *et al.* [4] constructed a sub-simhash index based on inverted tables and then encrypted it with the Paillier homomorphic encryption algorithm. Furthermore, instead of calculating the Euclidean distance, Guo *et al.* [5] calculated the lower bound of the Euclidean distance by using the mean and standard deviation of features. Then, they employed the Paillier homomorphic encryption algorithm to encrypt the mean and standard deviation of features to achieve the exact nearest neighbor search over encrypted images. Based on [5], Yang *et al.* [22] proposed an encrypted image search in the multi-user setting by encrypting the lower bound on the squared Euclidean distance with the distributed two trapdoors public-key cryptosystem [23].

Many schemes based on ASPE are proposed to avoid the high computational and communication overheads caused by homomorphic encryption. Yuan *et al.* [7] and Li *et al.* [10] designed two lightweight encrypted image search schemes by constructing an index tree with ASPE and clustering algorithms. In [8] and [9], Xia *et al.* used the Local Sensitive Hash (LSH) and ASPE to build an encrypted hash table for encrypted images similarity search in cloud computing. For security improvement, various enhanced ASPE algorithms are proposed. Based on the Learning with Errors (LWE) problem [24], Wang *et al.* [25] and Li *et al.* [26] used the enhanced ASPE algorithm to encrypt features, which ensures that features are secure against the known-background attacks. To protect the values and orders of similarity scores, Li *et al.* [27] and Song *et al.* [28] designed a privacy-preserving threshold-based image search scheme by using the matrix encryption techniques, respectively. In their schemes, each feature is expanded into a matrix instead of one or two vectors before encryption, which obtains security under Chosen-Plaintext Attack (CPA) model.

In addition to encrypting feature vectors with HE and ASPE algorithms, there are schemes to protect the privacy of feature vectors with the popular Inter Software Guard Extensions (SGX) and secure multi-party computation techniques. Schemes [29] and [30] utilized the SGX technique to achieve secure search over ciphertexts. The SGX decrypts encrypted images and feature vectors, computes the similarity over plaintexts, and returns the matched encrypted images. In [31], the secure multi-party computation technique is used to encrypt image feature vectors and achieve encrypted image search in multi-source settings. These schemes extract feature vectors locally before encrypting images to achieve retrieval. To reduce the local overhead, Xia *et al.* [32] encrypted the image locally and then uploaded it to the cloud server for extracting and retrieving secure features on the encrypted image. They proposed a novel Bag-of-Encrypted Words (BOEW) model based on the Bag-of-Visual Words (BOVW) model. BOEW extracts the secure local histograms from the encrypted images and then organizes them to a secure feature vector. After that, the Manhattan distances between feature vectors are computed on the cloud server side. In [33], Xia *et al.* applied the BOEW and Term Frequency–Inverse Document Frequency (TF-IDF) models to design a searchable image encryption scheme that is compatible with JPEG images.

To improve efficiency and security, we will propose an image search scheme based on Bloom filter. Compared with previous schemes, our scheme has the following advantages shown in TABLE I.

TABLE I
IMAGE SEARCH SCHEMES: A COMPARATIVE SUMMARY.

Schemes	Index Type	Index Protection	Resistible Attack	Search Complexity	Storage Overhead
[7]	Tree	ASPE	Broken	$O(\log n)$	$O(nv_1)$
[8]	Inverted	ASPE	Broken	$O(DB(w_q))$	$O(nv_1)$
[9]	Inverted	ASPE	Broken	$O(DB(w_q))$	$O(nv_1)$
[10]	Tree	ASPE	Broken	$O(\log n)$	$O(nv_1)$
[11]	Linear	ASPE	Broken	$O(n)$	$O(nv_1)$
[2]	Linear	HE	CPA	$O(n)$	$O(nv_2)$
[3]	Linear	HE	CPA	$O(n)$	$O(nv_2)$
[4]	Inverted	HE	CPA	$O(DB(w_q))$	$O(nv_2)$
[5]	Linear	HE	CPA	$O(n)$	$O(nv_2)$
[25]	Linear	LWE-ASPE	KBA	$O(n)$	$O(nv_1)$
[26]	Linear	LWE-ASPE	KBA	$O(n)$	$O(nv_1)$
[27]	Linear	Matrix	CPA	$O(n)$	$O(nv_1^2)$
[28]	Linear	Matrix	CPA	$O(n)$	$O(nv_1^2)$
Ours	BF	Hash	CKA2	$O(rkb)$	$O(rm)$

- "CPA": Chosen-Plaintext Attack; "KBA": Known-Background Attack; "CKA": Chosen-Keyword Attack.
- Security comparison: KBA-secure < CPA-secure < CKA2-secure, "CKA2-secure": adaptively secure under CKA.
- "n": the number of images; "$DB(w_q)$": the number of images with w_q; "k": the number of hash functions; "r": the number of Bloom filters.
- "v_1": the ciphertext size of ASPE encryption; "v_2": the ciphertext size of homomorphic encryption; "m": the length of Bloom filter.

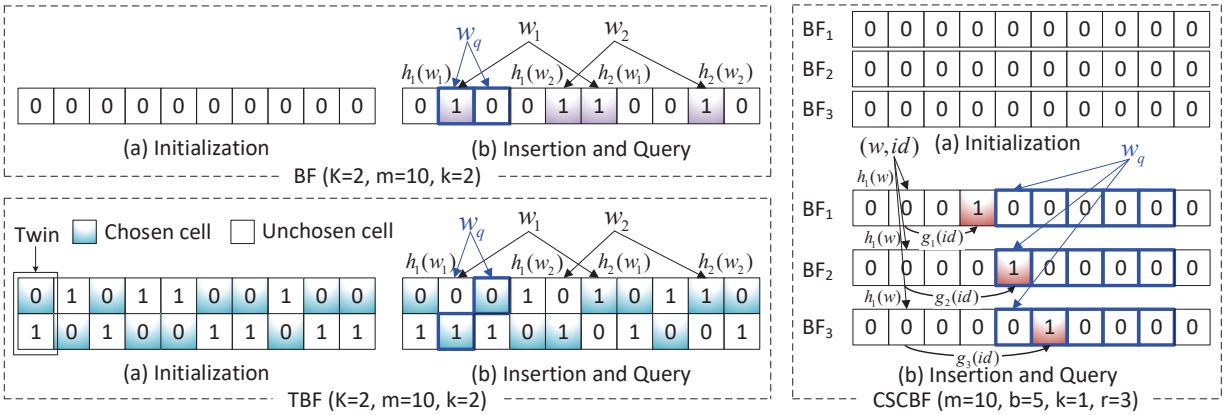

Fig. 2. The illustrations of BF, TBF and CSCBF.

III. Preliminaries

In this section, we will introduce Bloom Filter (BF) [13], Twin Bloom Filter (TBF) [18], and Circular Shift and Coalesce Bloom Filter (CSCBF) [20], which are used as basic technologies for our scheme.

A. Bloom Filter (BF)

As shown in Fig. 2, BF has m cells and each cell is initialized with 0. Given a keyword set $W = \{w_1, w_2, \cdots, w_K\}$ which contains K keywords and a hash function family $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), \cdots, h_k(\cdot)\}$ where each hash function hashes any keyword to $[0, m-1]$. To insert a keyword $w_\beta \in W$, BF computes the k locations $h_1(w_\beta), h_2(w_\beta), \cdots, h_k(w_\beta)$ and then sets the k cells $\mathsf{BF}[h_1(w_\beta)], \mathsf{BF}[h_2(w_\beta)], \cdots, \mathsf{BF}[h_k(w_\beta)]$ to 1. For any query keyword w_q, BF returns *True* if the k cells $\mathsf{BF}[h_1(w_q)], \mathsf{BF}[h_2(w_q)], \cdots, \mathsf{BF}[h_k(w_q)]$ (cells with blue lines in Fig. 2) are 1, and *False* otherwise.

For any keyword $w' \notin W$, BF may return *True* since $\mathsf{BF}[h_1(w)], \mathsf{BF}[h_2(w)], \cdots, \mathsf{BF}[h_k(w)]$ may be set to 1 by other keywords, which causes false positives. When m is very large, the false positive rate [34], [35] of BF is approximated as

$$\epsilon_{BF}(m,k,K) \approx \left(1 - \left(1 - \frac{1}{m}\right)^{k \cdot K}\right)^k \approx \left(1 - e^{-\frac{k \cdot K}{m}}\right)^k. \quad (1)$$

B. Twin Bloom Filter (TBF)

The values in BF are exposed, TBF [18] masks the values by a hash function and a random number to achieve adaptive security. As shown in Fig. 2, TBF has m twins and each twin has two cells to store 0 or 1. The hash function $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}$ and the random number δ determine which cell is chosen and initialized with 0, and the unchosen cell is initialized with 1. For a keyword $w_\beta \in W$, TBF uses $\mathcal{H}$ to get k twin locations $h_1(w_\beta), h_2(w_\beta), \cdots, h_k(w_\beta)$. Then, for k twins $\mathsf{TBF}(h_1(w_\beta)), \mathsf{TBF}(h_2(w_\beta)), \cdots, \mathsf{TBF}(h_k(w_\beta))$, TBF sets the chosen cell to 1 and the unchosen cell to 0 with $H(\cdot)$ and δ. For any query keyword w_q, TBF returns *True* if the k chosen cells in $\mathsf{TBF}[h_1(w_q)], \mathsf{TBF}[h_2(w_q)], \cdots, \mathsf{TBF}[h_k(w_q)]$ (cells with blue lines in Fig. 2) are 1, and *False* otherwise.

The m cells in TBF are set to 1 and the chosen cell in each twin is determined randomly. Thus, the probability of a polynomial-time adversary correctly guessing which cell is chosen in a twin is $\frac{1}{2} + \epsilon$. Here, ϵ is a negligible positive number. In addition, the false positive rate of TBF is the same as that of BF.

C. Circular Shift and Coalesce Bloom Filter (CSCBF)

To achieve higher storage utilization and efficiency as well as a lower false positive rate, Circular Shift and Coalesce Bloom Filter (CSCBF) was proposed in [20], which implements multi-set multi-relational approximate queries over the plaintext domain. As shown in Fig. 2, CSCBF includes $r \cdot m$ cells which are initialized with 0. To insert a keyword $w_\beta \in W$, the pair of (w_β, id) is inserted into CSCBF, where id is the identifier of file contained w_β. Moreover, CSCBF merges n files into b disjointly partitions by using a hash function $g(\cdot)$, and repeats r times to construct r BFs $\mathsf{BF}_1, \mathsf{BF}_2, \cdots, \mathsf{BF}_r$. For any query keyword w_q, CSCBF finds k locations $h_1(w_q), h_2(w_q), \cdots, h_k(w_q)$ in each BF. Then, for location $h_i(w_q)$ in BF_j where $i \in [1,k], j \in [1,r]$, CSCBF checks b follow up cells (cells with blue lines in Fig. 2) to get the candidate identifiers of files with w_q. Finally, the identifiers are obtained by computing the union and intersection of candidate identifier sets.

Note that, CSCBF constructs r BFs $\mathsf{BF}_1, \mathsf{BF}_2, \cdots, \mathsf{BF}_r$ for all files, while in BF and TBF each file has a BF or a TBF. Suppose the number of files is n, each file has a keyword set $W_\alpha \subseteq W$, and W_α contains $|W_\alpha|$ keywords. For a query keyword w_q in none of all files, the false positive rate of CSCBF is approximated as

$$\epsilon_{CSCBF} \approx \epsilon_{BF}(m,k,\sum_{\alpha=1}^{n}|W_\alpha|). \quad (2)$$

For a query keyword w_q in v files out of n files, the false positive rate of CSCBF is bounded as

$$\epsilon_{CSCBF} \lessapprox \left(1 - \left(1 - \epsilon_{BF}(m,k,\sum_{\alpha=1}^{n}|W_\alpha|)\right)\left(1 - \frac{1}{b}\right)^v\right)^r. \quad (3)$$

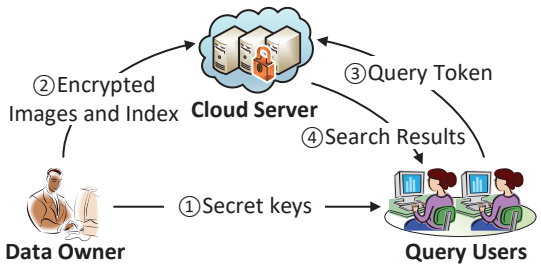

Fig. 3. System model of BFIS.

IV. Models and Design Goals

In this section, we describe our system model, threat model, problem definition, and design goals.

A. System Model

In our system, we consider an image search scenario, which consists of three parties, *i.e.*, a cloud server CS, a data owner DO, a set of query users QUs = $\{QU_1, QU_2, \cdots\}$. As shown in Fig. 3, the role of each entity is shown as follows.

1) Data Owner DO: DO generates keys and sends them to QUs. In addition, DO encrypts images, constructs a secure index, and finally sends them to CS.

2) Query Users QUs = $\{QU_1, QU_2, \cdots\}$: With the keys sent by DO, QU generates the query token for queried image and then sends the query token to CS.

3) Cloud Server CS: With the unlimited storage and powerful computation resources, CS provides storage service for DO and search service for QU. Upon receiving the query token from QU, CS honestly processes query and returns search results to QU.

As shown in Fig. 3, DO first generates and sends keys to QUs (step ①). Then, DO encrypts images via a symmetric encryption algorithm (*e.g.*, AES, chaos-based encryption), and constructs an index to achieve efficient search (step ②). When a certain QU wants to request a query, he/she generates a query token and then sends it to CS (step ③). Upon receiving the query token, CS first retrieves the index to obtain the identifiers of images similar to the query image, and then retrieves the encrypted image set to obtain the final results. Finally, CS returns the search results to QU for decryption (step ④).

B. Threat Model

In our system, DO is honest since he/she owns the images. Moreover, QUs are honest. Specifically, QUs will honestly issue the reasonable query and not collude with CS. As for CS, it is considered to be *honest-but-curious*. It will honestly store the encrypted images and index for DO and provide the search services for QUs. However, it may be curious about the plaintext of encrypted data such as encrypted images and index, as well as query tokens. Note that we are mainly concerned with data confidentiality, other leakages such as *access and search patterns* leakages are beyond the scope of this paper and will be discussed in future work.

C. Problem Definition

Suppose $IMG = \{img_1, img_2, \cdots, img_n\}$ is the image set of DO. QU selects an interesting image img_q as the query image. Then, CS computes the Euclidean distances between $img_\alpha \in IMG$ and img_q for $\alpha \in [1, n]$. According to the distances, CS finds out the images whose distance from img_q is within a certain range or smallest. In our scheme, to avoid computing distance, we study the keyword-based image search problem which is defined as follows.

Definition 1. *(Keyword-based Image Search). Each image $img_\alpha \in IMG$ is represented as a keyword set $W_\alpha \subseteq W$, where $\alpha \in [1, n]$ and $W = \{w_1, w_2, \cdots, w_K\}$ is a dictionary. Also, the query image m_q is represented as a keyword set $W_q \subseteq W$. Given a threshold $\tau \in [1, |W_q|]$, the search result R*

$$R = \{img_\alpha || W_\alpha \cap W_q| \geq \tau, \alpha \in [1, n]\}. \tag{4}$$

is returned, where $|\cdot|$ denotes the number of elements in set.

Give an example, suppose $W = \{w_1, w_2, w_3, w_4\}$, the keyword sets of three images img_1, img_2, img_3 and query image img_q are $W_1 = \{w_1, w_3\}$, $W_2 = \{w_1, w_2, w_3, w_4\}$, $W_3 = \{w_3, w_4\}$, and $W_q = \{w_1, w_2, w_3\}$, respectively. If $\tau = 2$, then CS returns the images img_1, img_2 since $|W_1 \cap W_q| = 2, |W_2 \cap W_q| = 3$. If $\tau = 3$, the image img_2 is returned. During the image search process, the *honest-but-curious* CS aims to obtain the plaintexts related to $IMG, W, \{W_1, W_2, \cdots, W_n\}, m_q, W_q$, while QU hopes to receive accurate search results as quickly as possible and keep m_q, W_q confidential since m_q may involve sensitive information.

D. Design Goals

Our design goals are to propose a secure and efficient image search scheme namely BFIS. Specifically, BFIS should satisfy the following requirements.

- *Adaptive security*: In the proposed BFIS, in addition to the images and query token should be privacy-preserving, the index should achieve adaptive security. That is, the adversary cannot distinguish the index simulated by the simulator.
- *Faster-than-linear efficiency*: To provide an efficient search service for QUs, the search results should be returned as soon as possible. Specifically, the search process should be faster than linear when deployed in large-scale datasets.
- *Low false positive rate*: The search results received by QUs should be as accurate as possible with minimal storage overhead. In other words, the false positive rate of the index should be low to ensure the validity of search results.

V. Our Proposed Scheme

In this section, we first design a novel index structure, namely Merged and Repeated Indistinguishable Bloom Filter (MRIBF). Then, with the MRIBF, we present a secure and efficient Bloom Filter-based Image Search scheme (BFIS). TABLE II gives some important notations used in this paper.

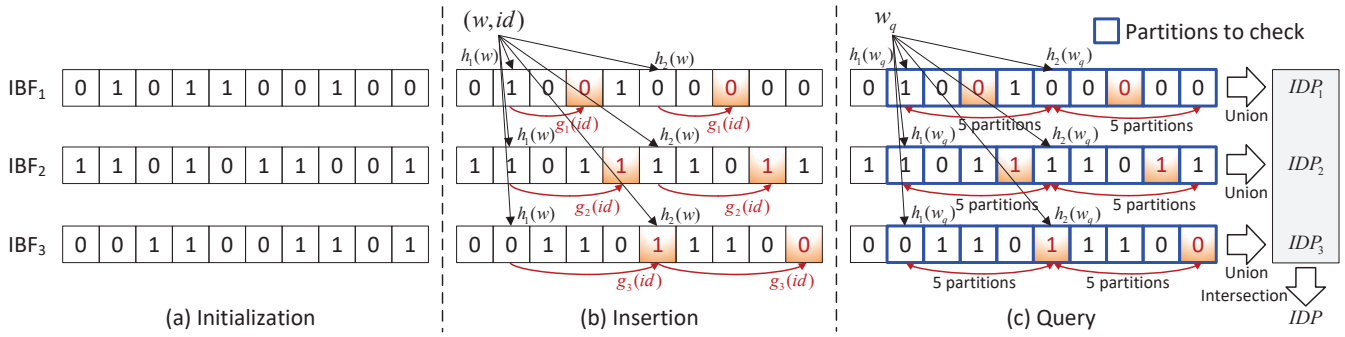

Fig. 4. An illustration of MRIBF.

TABLE II
NOTATIONS.

Notations	Descriptions
$[a, b]$	$\{a, a+1, \cdots, b\}$
$\|A\|$	The number of elements of set A
IMG	Image dataset, $IMG = \{img_1, img_2, \cdots, img_n\}$
n	The number of images in IMG
K	The number of clustering center vectors
$\{\vec{c}_1, \vec{c}_2, \cdots, \vec{c}_K\}$	Clustering center vectors
W	Dictionary, $W = \{w_1, w_2, \cdots, w_K\}$
W_α	Keyword set of img_α, $W_\alpha \subseteq W$
I	Generated index, $I = \{\mathsf{IBF}_i\}_{i=1}^r$
img_q	Query image of QU
$\vec{q}$	Feature vector of img_q
W_q	Keyword set of img_q
TK	Query token of img_q
$P_{j,t}$	The t-th partition of IBF_j
$IDP_{j,t}$	Identifiers contained in partition $P_{j,t}$

A. Main Idea of Our Scheme

In Section III, we have described the BF, TBF, and CSCBF index structures for data efficient search. However, they still suffer from the following limitations.

- *Waste storage space*. Since each file has a BF or TBF, the total number of BF or TBF increases with the total number of files. If BF or TBF's length is far larger than what needed to reduce the false positive rate, it will result in many cells being empty, which is a waste of storage space.
- *High false positive rate*. If BF or TBF's length is not large enough to save storage space, multiple data may be mapped to the same cell as the data increases. The more collisions when inserting data, the higher the false positive rate when querying data.
- *Expose the values in cells*. Although the storage overhead of TBF is twice that of BF, TBF masks the values in cells with a hash function and a random number, which implements adaptive security. However, BF and CSCBF expose the values in cells, which helps the adversary to infer the relationship between files.

In this paper, we will work on addressing the above problems. First, by modifying the initialization, insertion, and checking algorithms of TBF, we make the storage overhead of the modified TBF only half that of the original TBF without compromising security. The modified TBF is called Indistinguishable Bloom Filter (IBF), which remains the adaptive security. Then, based on IBF, we protect the values in

MRIBF's Constructions

• $\{\mathcal{H}, h_{k+1}(\cdot), H(\cdot), \mathcal{G}, \Delta\} \leftarrow \mathsf{MRIBF.Setup}(1^\kappa, k, m, b, r)$: Given the security parameter κ, the number of hash functions k, the length m of IBF, the number of partitions b, and the number of repetitions r, this algorithm generates the $k+1$ secret keys $SK_1, SK_2, \cdots, SK_{k+1}$. Then, a hash function family $\mathcal{H} = \{h_i(\cdot)\}_{i=1}^k$ is constructed using the key-hashed message authentication code (HMAC), where $h_i(\cdot) = \mathsf{HMAC}_{SK_i}(\cdot)\%m$. Besides, other hash functions $h_{k+1}(\cdot)$, $H(\cdot)$, and a hash function family $\mathcal{G} = \{g_j(\cdot)\}_{j=1}^r$ are constructed, where $h_{k+1}(\cdot) = \mathsf{HMAC}_{SK_{k+1}}(\cdot)$, $H(\cdot) = \mathsf{SHA1}(\cdot)\%2$, $g_j(\cdot) = \mathsf{SHA1}(\cdot)\%b$. At last, a set of random numbers $\Delta = \{\delta_j\}_{j=1}^r$ is generated.

• $\{\mathsf{IBF}_j\}_{j=1}^r \leftarrow \mathsf{MRIBF.Initi}(m, r, \mathcal{H}, h_{k+1}(\cdot), H(\cdot), \mathcal{G}, \Delta)$: This algorithm initializes r IBFs $\{\mathsf{IBF}_j\}_{j=1}^r$, and each IBF has m cells. Specifically, for $i \in [0, m-1]$, $j \in [1, r]$,

$$\mathsf{IBF}_j[i] = \begin{cases} 0, & H(h_{k+1}(i) \oplus \delta_j) = 0; \\ 1, & H(h_{k+1}(i) \oplus \delta_j) = 1. \end{cases} \tag{5}$$

• $\{\mathsf{IBF}_j\}_{j=1}^r \leftarrow \mathsf{MRIBF.Insert}((w, id), \{\mathsf{IBF}_j\}_{j=1}^r, m, r, \mathcal{H}, h_{k+1}(\cdot), H(\cdot), \mathcal{G}, \Delta)$: For a pair of keyword and identifier (w, id), this algorithm inserts w to the r IBFs $\mathsf{IBF}_1, \mathsf{IBF}_2, \cdots, \mathsf{IBF}_r$. Specifically, for $i \in [1, k]$, $j \in [1, r]$,

$$\mathsf{IBF}_j[h'_{i,j}(w)] = \begin{cases} 1, & H(h_{k+1}(h'_{i,j}(w)) \oplus \delta_j) = 0; \\ 0, & H(h_{k+1}(h'_{i,j}(w)) \oplus \delta_j) = 1, \end{cases} \tag{6}$$

where

$$h'_{i,j}(w) = (h_i(w) + g_j(id))\%m. \tag{7}$$

• $\{h'_{i,t}(w_q), h''_{i,t}(w_q)\}_{i=1,t=0}^{k,b-1} \leftarrow \mathsf{MRIBF.GenQ}(w_q, k, m, b, \mathcal{H})$: Given a query keyword w_q, for $i \in [1, k]$, $t \in [0, b-1]$, this algorithm computes the query request as

$$\begin{aligned} h'_{i,t}(w_q) &= (h_i(w_q) + t)\%m, \\ h''_{i,t}(w_q) &= h_{k+1}((h_i(w_q) + t)\%m). \end{aligned} \tag{8}$$

• $R \leftarrow \mathsf{MRIBF.Check}(\{h'_{i,t}(w_q), h''_{i,t}(w_q)\}_{i=1,t=0}^{k,b-1}, \{\mathsf{IBF}_j\}_{j=1}^r, H(\cdot), \Delta)$: With the query request, this algorithm searches each IBF. For IBF_j and $t \in [0, b-1]$, this algorithm checks

$$\mathsf{IBF}_j[h'_{i,t}(w_q)] \stackrel{?}{=} \begin{cases} 1, & H(h''_{i,t}(w_q) \oplus \delta_j) = 0; \\ 0, & H(h''_{i,t}(w_q) \oplus \delta_j) = 1. \end{cases} \tag{9}$$

If there exists t whose k values satisfy Eq. 9, then the identifiers of data with w_q may be in partition $P_{j,t}$. For partitions reporting the existence of w_q in IBF_j, this algorithm finds the identifiers contained in these partitions and then computes their union set IDP_j. For $\{\mathsf{IBF}_j\}_{j=1}^r$, the intersection of r sets are computed to obtain the final results IDP,

$$IDP \leftarrow IDP_1 \cap IDP_2 \cap \cdots \cap IDP_r. \tag{10}$$

Fig. 5. MRIBF's constructions.

CSCBF and propose a novel index which reduces the false positive rate, namely Merged and Repeated Indistinguishable Bloom Filter (MRIBF). Compared with BF, TBF, and CSCBF, MRIBF achieves efficient search with low storage, low false positive rate, and adaptive security. As shown in Fig. 4, the basic building block of MRIBF is an m-bit IBF. It is masked by a hash function H and a random number δ to achieve adaptive security. The unique partition function g_j maps n identifiers disjointly into b partitions to reduce the storage overhead. And r repetitions have the same $\mathcal{H}, h_{k+1}, H$ and different g_j, δ to reduce the false positive rate.

The constructions of MRIBF are shown in Fig. 5, which consists of five algorithms, namely $\mathsf{MRIBF.Setup}(\cdot)$, $\mathsf{MRIBF.Initi}(\cdot)$, $\mathsf{MRIBF.Insert}(\cdot)$, $\mathsf{MRIBF.GenQ}(\cdot)$, as well as $\mathsf{MRIBF.Check}(\cdot)$. At first, hash functions and random numbers are generated in $\mathsf{MRIBF.Setup}(\cdot)$. Then, r m-bit IBFs $\mathsf{IBF}_1, \mathsf{IBF}_2, \cdots, \mathsf{IBF}_r$ are initialized in $\mathsf{MRIBF.Initi}(\cdot)$. Next, $\mathsf{MRIBF.Insert}(\cdot)$ inserts the pair of (w, id) into r IBFs, where w is a keyword and id is the identifier of file with w. For any query keyword w_q, $\mathsf{MRIBF.GenQ}(\cdot)$ generates the query token for search. At last, $\mathsf{MRIBF.Check}(\cdot)$ searches r IBFs and obtains identifiers of files with w_q.

Example. In Fig. 4, we give an example of MRIBF with $k = 2$, $m = 10$, $b = 5$ and $r = 3$. According to the $\mathsf{MRIBF.Setup}(\cdot)$ and $\mathsf{MRIBF.Initi}(\cdot)$ algorithms, we initialize 3 IBFs $\mathsf{IBF}_1, \mathsf{IBF}_2, \mathsf{IBF}_3$ as shown in Fig. 4(a). Then, given a pair of (w, id), we insert it into $\mathsf{IBF}_1, \mathsf{IBF}_2, \mathsf{IBF}_3$ with the algorithm $\mathsf{MRIBF.Insert}(\cdot)$. As shown in Fig. 4(b), $g_1(id), g_2(id), g_3(id)$ are three shift ways in $\mathsf{IBF}_1, \mathsf{IBF}_2, \mathsf{IBF}_3$, respectively. Next, with the query keyword w_q, we find its locations in IBFs by computing $h_1(w_q), h_2(w_q)$. In Fig. 4(c), the cells with blue lines are needed to check whether they satisfy Eq. 9. For IBF_1 and $t = 2$, $\mathsf{IBF}_1[3]$ satisfies Eq. 9. Thus, $P_{1,2}$ may contain the identifiers of data with w_q. Similarly, for IBF_2 and IBF_3, two partitions $P_{2,3}$ and $P_{3,4}$ are obtained. Finally, we find the identifiers IDP_1, IDP_2, IDP_3 contained in $P_{1,2}, P_{2,3}, P_{3,4}$, and compute the intersection of these identifiers to obtain the final results $IDP = IDP_1 \cap IDP_2 \cap IDP_3$.

B. Secure and Efficient Bloom Filter-based Image Search Scheme (BFIS)

Our MRIBF index can be applied for various data search. Here, we give a secure and efficient Bloom filter-based image search scheme (BFIS) to show the MRIBF applications for image search.

To match the input of MRIBF, we use keywords to represent images as in existing image search schemes based on Bag-of-Visual-Word (BOVW) model [36]–[38]. The BOVW model first clusters the feature vectors of all images into a dictionary and then counts the frequency of each keyword to obtain a feature vector for each image. Similarly, we extract feature vectors from n images and cluster them into K clusters. The K clustering center vectors are regarded as K keywords and form a dictionary $W = \{w_1, w_2, \cdots, w_K\}$. According to the clustering results, each feature vector is represented as a keyword, and each image is represented as a keyword set $W_\alpha \subseteq W (\alpha \in [1, n])$. Then, we propose the secure and efficient Bloom filter-based image search scheme (BFIS), which consists of four phases, *i.e.*, system initialization, data outsourcing, query token generation, and query processing. Details are shown as follows.

1) System Initialization: In the system initialization phase, DO is responsible for generating keys. First, he/she determines κ, k, m, b, r and runs the $\mathsf{MRIBF.Setup}(\cdot)$ algorithm to generates $\mathcal{H} = \{h_1(\cdot), h_2(\cdot), \cdots, h_k(\cdot)\}$, $h_{k+1}(\cdot)$, $H(\cdot)$, $\mathcal{G} = \{g_1(\cdot), g_2(\cdot), \cdots, g_r(\cdot)\}$, and $\Delta = \{\delta_1, \delta_2, \cdots, \delta_r\}$. Then, DO sends P_1 to CS and P_2 to QU.

$$P_1 = \{r, b, k, H(\cdot), \mathcal{G}, \Delta\}, P_2 \quad = \{m, b, \mathcal{H}, h_{k+1}(\cdot)\}.$$

At last, DO initializes $\{\mathsf{IBF}_i\}_{i=1}^r$ with the $\mathsf{MRIBF.Initi}(\cdot)$ algorithm.

2) Data Outsourcing: In the data outsourcing phase, DO builds an index based on image features and outsources the index to CS.

- `Step 1: Generate the pair of` (w_β, IDs) `for keyword` $w_\beta \in W$ `where` $\beta \in [1, K]$. At first, DO extracts the feature vector set F_α of each image $img_\alpha \in IMG$, where $\alpha \in [1, n]$. Then, with the K-means clustering method, DO clusters all feature vector sets $F = \{F_1, F_2, \cdots, F_n\}$ to obtain K clustering center vectors $\vec{c}_1, \vec{c}_2, \cdots, \vec{c}_K$. Each feature vector in F can be represented by its nearest clustering center vector. We regard K cluster centers as K keywords. Thus, the dictionary W can be represented as

  $$W = \{w_1, w_2, \cdots, w_K\} = \{\vec{c}_1, \vec{c}_2, \cdots, \vec{c}_K\},$$

  and the feature vector set F_α can be represented as a keyword set $W_\alpha \subseteq W$. For each keyword $w_\beta \in W$, DO generates the pair of (w_β, IDs), where IDs denotes the identifiers of images contained w_β.
- `Step 2: Insert` (w_β, IDs) `into` $\{\mathsf{IBF}_i\}_{i=1}^r$ `where` $\beta \in [1, K]$. For each (w_β, IDs) pair, $\beta \in [1, K]$, DO inserts it into $\{\mathsf{IBF}_i\}_{i=1}^r$ by recalling the $\mathsf{MRIBF.Insert}(\cdot)$ algorithm. First, DO computes the k hash values $h_1(w_\beta), h_2(w_\beta), \cdots, h_k(w_\beta)$ with $\mathcal{H}$. Then, for each $id \in IDs$, $j \in [1, r]$ and $i \in [1, k]$, DO computes Eq. 7 with g_j to obtain the insertion location $\mathsf{IBF}_j[(h_i(w_\beta) + g_j(id))\%m]$. After that, DO computes Eq. 6 with H and δ_j to set the cell's value. For $\beta \in [1, K]$, when all (w_β, IDs) pairs are inserted into $\{\mathsf{IBF}_i\}_{i=1}^r$, the building of search index

  $$I = \{\mathsf{IBF}_i\}_{i=1}^r$$

  is finished. At last, DO sends I to CS.

3) Query Token Generation: For the query image m_q, QU first extracts its feature vector set $F_q = \{\vec{f}_1, \vec{f}_2, \cdots, \vec{f}_{n_q}\}$, where n_q is the number of feature vectors. Then, QU finds the nearest clustering center vector for each $\vec{f}_i \in F_q (i \in [1, n_q])$ by calculating the square of Euclidean distances

$$||\vec{c}_1 - \vec{f}_i||^2, ||\vec{c}_2 - \vec{f}_i||^2, \cdots, ||\vec{c}_K - \vec{f}_i||^2.$$

After that, QU represents F_q as a keyword set $W_q \subseteq W$ by representing each $\vec{f}_i \in F_q (i \in [1, n_q])$ as its nearest clustering vector. For each query keyword $w_\gamma \in W_q (\gamma \in [1, |W_q|])$, QU

recalls the $\mathsf{MRIBF.GenQ}(\cdot)$ algorithm to generate the query token TK_γ as

$$TK_\gamma = \{h'_{i,t}(w_\gamma)h''_{i,t}(w_\gamma)\}_{i=1,t=0}^{k,b-1},$$

where $h'_{i,t}(w_\gamma)$ and $h''_{i,t}(w_\gamma)$ are computed with Eq. 8. Finally, $TK = \{TK_\gamma\}_{\gamma=1}^{|W_q|}$ is sent to CS.

4) Query Processing: Upon receiving TK from QU, CS recalls the $\mathsf{MRIBF.Check}(\cdot)$ algorithm $|W_q|$ times to get $|W_q|$ results $R_1, R_2, \cdots, R_{|W_q|}$. Then, the identifiers belong to τ results $\{R'_1, R'_2, \cdots, R'_\tau\} \subseteq \{R_1, R_2, \cdots, R_{|W_q|}\}$ form the final result R. Specifically, given a query token $TK_\gamma \in TK$, for each $t \in [0, b-1]$ in $\mathsf{IBF}_j (j \in [1, r])$, CS checks whether $\mathsf{IBF}_j[h'_{1,t}(w_\gamma)], \mathsf{IBF}_j[h'_{2,t}(w_\gamma)], \cdots, \mathsf{IBF}_j[h'_{k,t}(w_\gamma)]$ satisfy Eq. 9. If there exists t makes Eq. 9 hold, then the identifiers of images with w_γ may be in partition $P_{j,t}$. According to the $\mathsf{MRIBF.Check}(\cdot)$ algorithm, CS finally obtains the search result $R_\gamma = IDP_\gamma$ for query token TK_γ. Similarly, for $TK = \{TK_\gamma\}_{\gamma=1}^{|W_q|}$, CS gets $|W_q|$ search results $R_1, R_2, \cdots, R_{|W_q|}$. The final search result R is obtained by computing

$$R = \{id | R'_1 \cap R'_2 \cap \cdots \cap R'_\tau \cap id \neq \varnothing\}$$

for each $id \in \{R_1 \cup R_2 \cup \cdots \cup R_{|W_q|}\}$ and $\{R'_1, R'_2, \cdots, R'_\tau\} \subseteq \{R_1, R_2, \cdots, R_{|W_q|}\}$.

Remark. In our BFIS, we focus on how to construct and retrieve the index I. According to the search result R, CS can find the corresponding images in the encrypted image dataset and return them to QU. Since the image encryption and decryption processes are independent of index construction and retrieval processes, and they can be implemented with existing methods such as AES and chaos-based encryption, we do not introduce image encryption and decryption in the proposed BFIS.

VI. Theoretical Analysis

In this section, we analyze the false positive rate and security of our scheme.

A. False Positive Rate Analysis

The false positive rate of our scheme BFIS is caused by the index structure MRIBF. For simplicity, we assume that each image has one keyword.

Theorem 1. *Assume that a query keyword w_q of query image img_q is the same as that of the v images out of n images. For an image img_x having a different keyword from the query image, the false positive rate ϵ_{MRIBF} of MRIBF returns img_x is*

$$\epsilon_{MRIBF} = \left(1 - (1 - \epsilon_{BF}(m, k, K))\left(1 - \frac{1}{b}\right)^v\right)^r,$$

where $r \geq 2$ and $\epsilon_{BF}(m, k, K)$ (shown in Eq. (1)) is the false positive rate of BF with m bits, k hash functions and K keywords inserted.

Proof. The probability of any two identifiers inserted into the same partition is $Pr(g(id_1) = g(id_2)) = \frac{1}{b} \cdot \frac{1}{b} \cdot b = \frac{1}{b}$. We analyze two cases as follows.

- *Case 1:* The probability that x is not in the same partition as the identifiers of v images is $\left(1 - \frac{1}{b}\right)^v$.
- *Case 2:* the probability that x is in the same partition as the identifiers of v images is $1 - \left(1 - \frac{1}{b}\right)^v$.

For one repetition, a false positive w_q occurs must belong to one of these two cases. The false positive must occurs in *Case 2*. While in *Case 2*, the false positive may be occur when the Bloom filter incorrectly reports it with a probability $\epsilon_{BF}(m, k, K)\left(1 - \frac{1}{b}\right)^v$. Therefore, for one repetition, the false positive rate is

$$1 - \left(1 - \frac{1}{b}\right)^v + \epsilon_{BF}(m, k, K)\left(1 - \frac{1}{b}\right)^v.$$

For r repetitions, the total false positive rate is

$$\begin{aligned}\epsilon &= \left(1 - \left(1 - \frac{1}{b}\right)^v + \epsilon_{BF}(m, k, K)\left(1 - \frac{1}{b}\right)^v\right)^r \\ &= \left(1 - (1 - \epsilon_{BF}(m, k, K))\left(1 - \frac{1}{b}\right)^v\right)^r.\end{aligned}$$

□

B. Security Analysis

We prove that our BFIS is adaptively secure under the *chosen keyword attack* (CKA) model [19]. We first define the following two leakage functions.

- $\mathcal{L}_1(I, IMG)$: Given the index I and data set IMG, this function outputs the size pattern, which contains the size of IBF, the number of images, and the partitions of image identifiers.
- $\mathcal{L}_2(I, IMG, q_1, \cdots, q_i)$: Given the index I, the data set IMG, and a series of queries $q_1, \cdots, q_i$, this function outputs the search pattern, which includes the information about whether the same query was performed before or not, and the access pattern, which includes the information about which images are returned for $q_1, \cdots, q_i$.

Based on $\mathcal{L}_1(I, IMG)$ and $\mathcal{L}_2(I, IMG, q_1, \cdots, q_i)$, we define the **Real** world and **Ideal** world as follows.

- **Real.** In the **Real** world, there are a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{B}$. First, the challenger $\mathcal{B}$ initializes the system in the way shown in the **System Initialization** phase. Then, the adversary $\mathcal{A}$ selects the image set IMG and sends it to $\mathcal{B}$. According to the steps in **Images Outsourcing** phase, $\mathcal{B}$ generates the index I and returns it to $\mathcal{A}$. Based on I, $\mathcal{A}$ chooses a query image $m_{q,1}$ for $\mathcal{B}$ and then receives a query token TK_1 which is generated with the methods in **Query Token Generation** phase. After that, $\mathcal{A}$ chooses a series of query images $m_{q,2}, \cdots, m_{q,u}$ and repeats the above processes in a polynomial time. Finally, $\mathcal{A}$ obtains $(I, TK_1, TK_2, \cdots, TK_u)$.
- **Ideal.** In the **Ideal** world, there are a PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ with leakage $\mathcal{L}_1$ and $\mathcal{L}_2$. First, the adversary $\mathcal{A}$ selects the image set IMG. Given $\mathcal{L}_1$, a simulator $\mathcal{S}$ generates and sends the index I^* to $\mathcal{A}$. Then, $\mathcal{A}$ chooses an image as the query image $m_{q,1}$. $\mathcal{S}$ generates

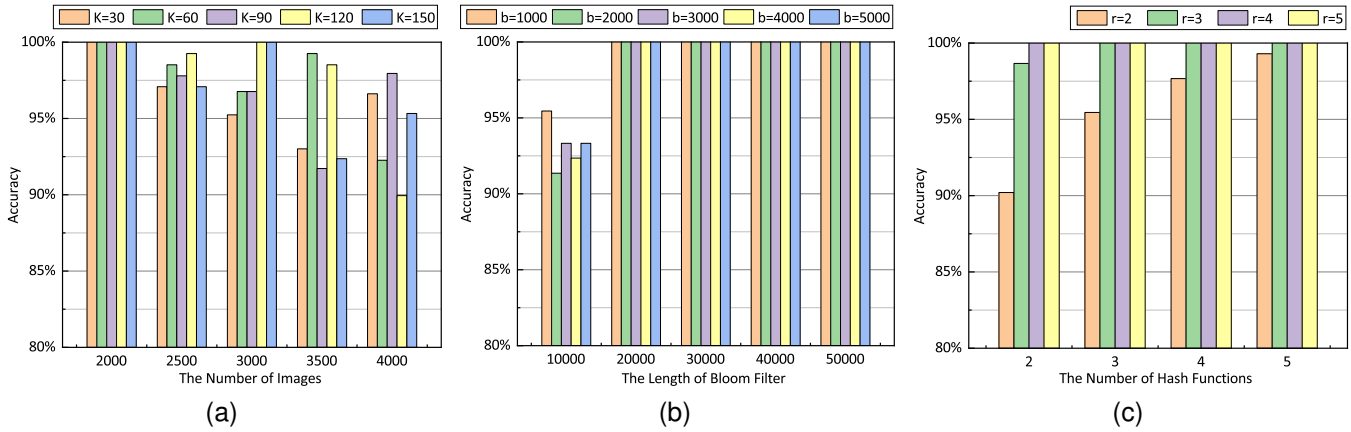

Fig. 6. Accuracy of BFIS.

a token TK_1^* for $m_{q,1}$ according to $\mathcal{L}_2$. Next, $\mathcal{A}$ chooses a series of query images $m_{q,2}, \cdots, m_{q,u}$ and repeats the above processes in a polynomial time. Finally, $\mathcal{A}$ obtains $(I^*, TK_1^*, TK_2^*, \cdots, TK_u^*)$.

Based on the **Real** world and **Ideal** world, we prove that BFIS is IND-CKA2 $(\mathcal{L}_1, \mathcal{L}_2)$-secure against an adaptive adversary.

Theorem 2. *BFIS is IND-CKA2 $(\mathcal{L}_1, \mathcal{L}_2)$-secure against an adaptive adversary.*

Proof. First, we construct a simulator $\mathcal{S}$ that can build a simulated index I^* based on the $\mathcal{L}_1(I, IMG)$. Specifically, it simulates the encrypted images $IMG^* = \{img_1^*, img_2^*, \cdots, img_n^*\}$ using AES or chaos-based encryption algorithms, the size of image set n and the size of each ciphertext. Then, $\mathcal{S}$ sets up r m-bit IBFs. For each cell, $\mathcal{S}$ flips a coin to decide the value stored in the cell. $\mathcal{S}$ also stores these coin values locally. Finally, $\mathcal{S}$ sends the simulated index I^* to $\mathcal{A}$.

Next, we show how $\mathcal{S}$ simulates queries on I^* based on $\mathcal{L}_2(I, IMG, q_1, \cdots, q_i)$. After receiving a query q_i, $\mathcal{S}$ knows whether q_i has been searched before according to the revealed *search pattern*. If it has been searched, $\mathcal{S}$ returns the same query token TK_i^* to $\mathcal{A}$. Otherwise, $\mathcal{S}$ generates a new query token TK_i^* as follows. According to the partitions $IDP_1, IDP_2, \cdots, IDP_r, IDP_{q_i}$ revealed by *access pattern* in $\mathcal{L}_2(I, IMG, q_1, \cdots, q_i)$, $\mathcal{S}$ finds $r \cdot k$ locations in $\mathsf{IBF}_1, \mathsf{IBF}_2, \cdots, \mathsf{IBF}_r$ so that these locations make Eqs. 9 and 10 hold. To simulate the hash function $h_{k+1}(\cdot)$, $\mathcal{S}$ selects m strings $s_1, s_2, \cdots, s_m$ for m cells such that the values of $H(\cdot)$ are the same as the stored coin values. Then, $\mathcal{S}$ returns $k \cdot b$ locations and their $k \cdot b$ strings as the query token TK_i^* to $\mathcal{A}$.

According to the simulation process, a probabilistic polynomial time adversary $\mathcal{A}$ cannot distinguish (I, TK_i) and (I^*, TK_i^*) since a pseudo-random function is computationally indistinguishable from a random function [39]. Thus, MRIBF is IND-CKA2 secure against an adaptive adversary.

□

VII. Performance Evaluation

We evaluate the performance of our BFIS in terms of accuracy, data outsourcing, query token generation, and query processing. In addition, we compare the performance of BFIS with other Bloom filters.

A. Experimental Setting

We conduct experiments on an Ubuntu 18.04 Server with 3.60GHz Intel(R) Core(TM) i7-6500K CPU by using Python. We randomly select 40,50,60,70,80 images from the first 50 categories in the real-world dataset *Caltech256* [40] to form 2000,2500,3000,3500,4000 images. Based on these images, a series of experiments are conducted. For simplicity, we extract the global feature vectors by using the pre-trained CNN model ResNet and reduce the dimension to 128 by using PCA (Principal Component Analysis) technology. In addition, we specify HMAC as SHA256.

B. Experimental Results

In this subsection, we will display the experimental results of accuracy, data outsourcing, query token generation, and query processing.

1) Accuracy: We compute the accuracy as

$$\frac{The\ number\ of\ false\ positive\ in\ returned\ images}{The\ number\ of\ returned\ images}.$$

The accuracy is related to n, K, m, b, k, r.

- *The parameters n and K:* Fig. 6a plots the accuracy in different n and K. The parameters are set as $n = \{2000, 2500, 3000, 3500, 4000\}$, $K = \{30, 60, 90, 120, 150\}$, $m = 20000$, $k = 3$, $r = 3$, $b = 2000$. From this figure, we can find that the accuracy decreases slightly as n increases. When k becomes larger, the accuracy will be higher. It is obvious that the decrease and increase are small.
- *The parameters m and b:* In Fig. 6b, we give the accuracy varying with m and b. In this experiment, the parameters are set as $m = \{10000, 20000, 30000, 40000, 50000\}$, $b = \{1000, 2000, 3000, 4000, 5000\}$, $n = 2000$, $K = 90$, $k = 3$, $r = 3$. When the length of Bloom filter m exceeds 20000, the accuracy can reach 100%. Meanwhile, the accuracy becomes higher as b increases. This is because the larger m and b are, the lower the probability of collision when inserting data and the lower the false positive rate will be.

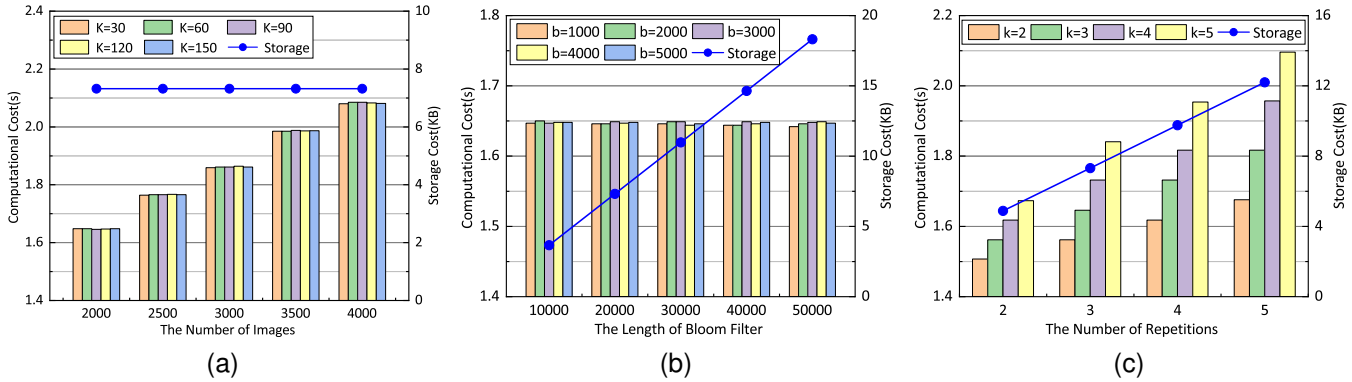

Fig. 7. Index generation cost of BFIS in **Data Outsourcing** phase.

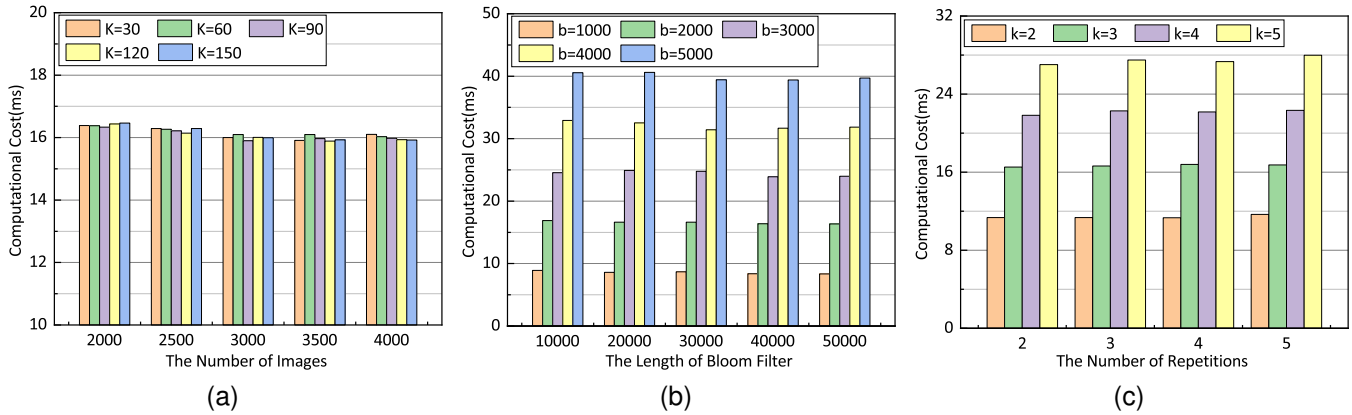

Fig. 8. Query token generation cost of BFIS in **Query Token Generation** phase.

• *The parameters k and r:* In Fig. 6c, we show the accuracy varying with k and r. We set parameters as $k = \{2, 3, 4, 5\}$, $r = \{2, 3, 4, 5\}$, $n = 2000$, $K = 90$, $m = 20000$, $b = 2000$. With the increase of k and r, the accuracy becomes higher. When $k = 3$, $r = 3$, the accuracy will reach 100%.

2) Data Outsourcing: We give the cost of index generation in data outsourcing phase. It is affected by parameters n, K, m, b, k, r.

• *The parameters n and K:* In Fig. 7a, we display the computational and storage cost versus n and K. The parameters are set as $n = \{2000, 2500, 3000, 3500, 4000\}$, $K = \{30, 60, 90, 120, 150\}$, $m = 20000$, $k = 3$, $r = 3$, $b = 1000$. From this figure, we can see that the computational cost of index generation increases with the increase of n but remains almost constant for K. Moreover, the storage cost is also unchanged since m and r are fixed.

• *The parameters m and b:* In Fig. 7b, we give the cost of data outsourcing varying with m and b. We set parameters as $m = \{10000, 20000, 30000, 40000, 50000\}$, $b = \{1000, 2000, 3000, 4000, 5000\}$, $n = 2000$, $K = 90$, $k = 3$, $r = 3$. From this figure, we can see that the computational cost is almost unchanged as m and b increase. However, the storage cost increases linearly with the increase of m.

• *The parameters k and r:* In Fig. 7c, we plot the cost of data outsourcing varying with k and r. We set parameters as $k = \{2, 3, 4, 5\}$, $r = \{2, 3, 4, 5\}$, $n = 2000$, $K = 90$, $m = 20000$, $b = 2000$, and we can see that the computational cost of data outsourcing increases with the increase of k and r. This is because the increase of k and r leads to more data insertion operations. Since r increases from 2 to 5, the storage cost increases from 4.88KB to 12.2KB.

3) Query Token Generation: As described in Section V-B, the computational cost of query token generation is related to K, b, k. Thus, we set $K = \{30, 60, 90, 120, 150\}$, $b = \{1000, 2000, 3000, 4000, 5000\}$, $k = \{2, 3, 4, 5\}$. Also, we give the computational cost related to n, m and r.

• *The parameters n and K:* In Fig. 8a, we plot the computational cost of query token generation varying with K and n while fixing $m = 20000$, $b = 2000$, $k = 3$, $r = 3$. From this figure, we can find that changes in n and K cause almost no change in computational cost, indicating that the main factors affecting the change in query token generation computational cost are b and k.

• *The parameters m and b:* Fig. 8b shows the computational cost varying with m and b. In this experiment, $n = 2000$, $K = 90$, $k = 3$, $r = 3$. When m increases from 10000 to 50000, the computational cost remains unchanged. While b increases from 1000 to 5000, the computational cost increases significantly. This is because the values of $k \cdot b$ cells are calculated when generating the query token.

• *The parameters k and r:* In Fig. 8c, we give the computational cost versus k and r for $n = 2000$, $K = 90$, $m = 20000$,

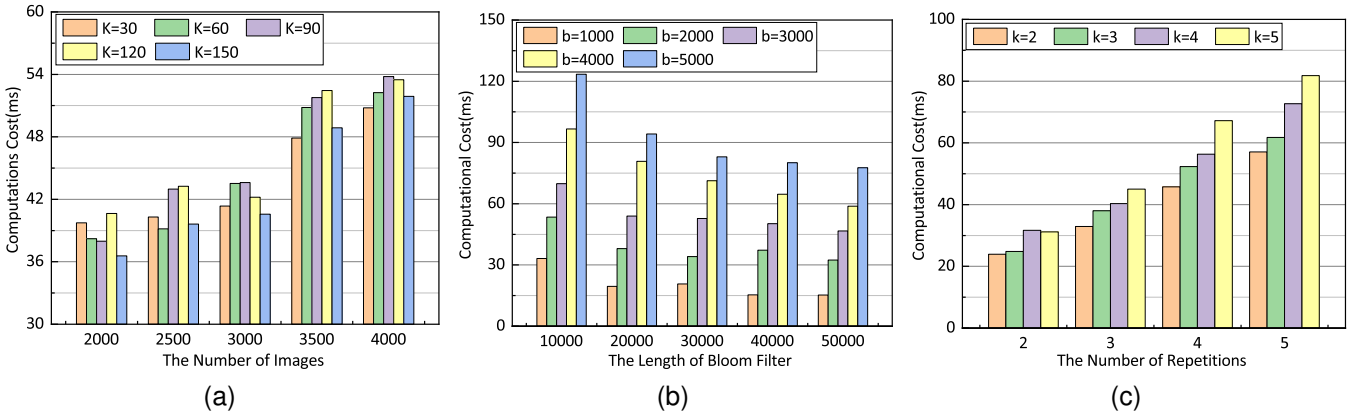

Fig. 9. Query processing cost of BFIS in **Query Processing** phase.

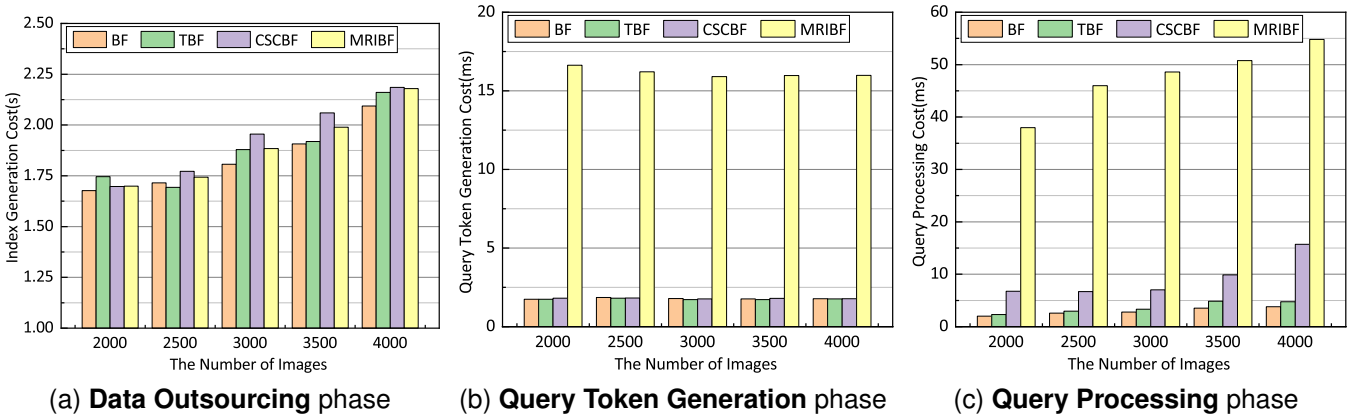

Fig. 10. Compared with other filters.

$b = 2000$. It can be seen that the change in r does not affect the cost of query token generation. When k varies from 2 to 5, the cost also becomes larger.

4) Query Processing: As described in Section V-B, the computational cost of query processing is related to parameters n, K, m, b, k, r.

• *The parameters n and K:* The computational cost of query processing varying with n and K is displayed in Fig. 9a. We set parameters as $n = \{2000, 2500, 3000, 3500, 4000\}$, $K = \{30, 60, 90, 120, 150\}$, $m = 20000$, $k = 3$, $r = 3$, $b = 1000$. In this figure, the cost increases as n increases. However, the impact of K on the cost is not significant since K affects the cost of computing the union and intersection sets. The set operation is very efficient and has a small cost in the overall query processing.

• *The parameters m and b:* Fig. 9b shows the computational cost of query processing varying with m and b. We set parameters as $m = \{10000, 20000, 30000, 40000, 50000\}$, $b = \{1000, 2000, 3000, 4000, 5000\}$, $n = 2000$, $K = 90$, $k = 3$, $r = 3$. We can see from Fig. 9b that the larger b is, the more cost will be required. While the cost varying with m is the opposite. This is because there require fewer operations for intersection and union of partitions as m increases.

• *The parameters k and r:* In Fig. 9c, we present the computational cost of query processing versus k and r when $k = \{2, 3, 4, 5\}$, $r = \{2, 3, 4, 5\}$, $n = 2000$, $K = 90$, $m = 20000$, $b = 2000$. Obviously, the increase in k and r introduces more computational cost since there are more cells to check.

5) Compared with Other Filters: In Fig. 10, we compare the performance of our MRIBF (Merged and Repeated Indistinguishable Bloom Filter) with BF (Bloom Filter), TBF (Twin Bloom Filter) and CSCBF (Circular Shift and Coalesce Bloom Filter) in terms of data outsourcing, query token generation, and query processing. We set the parameters are $n = \{2000, 2500, 3000, 3500, 4000\}$, $K = 90$, $m = 20000$, $b = 2000$, $k = 3$, $r = 3$. Since the accuracy of the four filters is 100%, we do not give a figure to show the results. The cost of index generation in the **Data Outsourcing** phase is shown in Fig. 10a. As the total number of images increases, the cost of four filters also increases. When we fix the total number of images, since the clustering operation has a great effect on the cost than the hash operation, the index generation cost of four filters is similar. During the **Query Token Generation** phase, MRIBF has significantly more cost as shown in Fig. 10b. This is because MRIBF needs to do $2kb$ hashes, while BF and CSCBF only need to do k hashes, and TBF only needs to do $2k$ hashes. In the **Query Processing** phase, because MRIBF needs to do kbr XOR and hash operations, the cost is more than other filters. In order to reduce the query processing cost, XOR and hash operations can be performed by QU.

However, this increases the query token generation cost. In this paper, to reduce the burden of terminal user QU as much as possible, we leave the XOR and hash operations to the CS to complete. In addition, although MRIBF increases significantly, its computational cost is still at the millisecond level and its storage cost will not increase proportionally with the total number of images, which is acceptable for terminal users.

6) Compared with Other Schemes: In TABLE III, we compare our BFIS with two other image search schemes under the different total numbers of images. One is the scheme with matrix encryption [28], and the other is the scheme with homomorphic encryption [3]. Without loss of generality, we replace the multi-key homomorphic encryption in [3] with the most classic Paillier homomorphic encryption. The parameters are set as $n = \{2000, 3000, 4000\}, d = 128, K = 90, m = 20000, b = 2000, k = 3, r = 3$. The computational cost of all three schemes increases as the total number of images increases. Since BFIS clusters image features with K-means clustering, the computational cost of index generation is higher than that of [28]. However, the storage cost of BFIS in index generation is minimal and reduced by at least a factor of 10^4 and 10^3 compared with [28] and [3]. In addition, BFIS also requires the minimal computational cost in query processing. These results indicate that BFIS has a obvious advantages in retrieval.

TABLE III
COMPARISON OF VARIOUS SCHEMES.

Computational and Storage Cost	n	BFIS	Matrix Encryption [28]	Homomorphic Encryption [3]
Index Generation	2000	1.69s 7.32KB	0.47s 125MB	8.04h 31.25MB
	3000	1.78s 7.32KB	0.69s 187.5MB	12.34h 46.875MB
	4000	2.08s 7.32KB	0.83s 250MB	16.16h 62.5MB
Query Processing	2000	0.03s	0.12s	13.53min
	3000	0.04s	0.21s	20.25min
	4000	0.05s	0.28s	27.18min

VIII. CONCLUSION

In this paper, we first design a merged and repeated indistinguishable Bloom filter index MRIBF. Then, based on MRIBF, we propose a secure and efficient image search scheme (BFIS). Compared with the previous schemes, our BFIS achieves higher security and efficiency. Detailed analysis shows that BFIS is really adaptively secure under chosen keyword attacks. In addition, we implemented BFIS in Python and evaluated its performance on the real-world dataset. Experimental results show that BFIS is efficient and feasible. In the future, we aim to promote BFIS from two aspects: (1) compressing the length of the filter via an XOR operation; (2) hiding search patterns and access patterns.

REFERENCES

[1] H. Arne, "Volume of data/information created worldwide from 2010 to 2024," *Statista*, 2020.

[2] Y. Zhang, L. Zhuo, Y. Peng, and J. Zhang, "A secure image retrieval method based on homomorphic encryption for cloud computing," in *2014 19th International Conference on Digital Signal Processing*. IEEE, 2014, pp. 269–274.

[3] L. Zhang, T. Jung, K. Liu, X.-Y. Li, X. Ding, J. Gu, and Y. Liu, "Pic: Enable large-scale privacy preserving content-based image search on cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3258–3271, 2017.

[4] M. Li, M. Zhang, Q. Wang, S. S. Chow, M. Du, Y. Chen, and C. Lit, "Instantcryptogram: Secure image retrieval service," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2222–2230.

[5] C. Guo, S. Su, K.-K. R. Choo, and X. Tang, "A fast nearest neighbor search scheme over outsourced encrypted medical images," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 514–523, 2018.

[6] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 139–152.

[7] J. Yuan, S. Yu, and L. Guo, "Seisa: Secure and efficient encrypted image search with access control," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2083–2091.

[8] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594–2608, 2016.

[9] Z. Xia, N. N. Xiong, A. V. Vasilakos, and X. Sun, "Epcbir: An efficient and privacy-preserving content-based image retrieval scheme in cloud computing," *Information Sciences*, vol. 387, pp. 195–204, 2017.

[10] X. Li, Q. Xue, and M. C. Chuah, "Casheirs: Cloud assisted scalable hierarchical encrypted based image retrieval system," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[11] Y. Li, J. Ma, Y. Miao, Y. Wang, T. Yang, X. Liu, and K.-K. R. Choo, "Traceable and controllable encrypted cloud image search in multi-user settings," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2936–2948.

[12] R. Li, A. X. Liu, Y. Liu, H. Xu, and H. Yuan, "Insecurity and hardness of nearest neighbor queries over encrypted data," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1614–1617.

[13] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[14] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM 2014-IEEE conference on computer communications*. IEEE, 2014, pp. 2112–2120.

[15] 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, no. 12, pp. 2706–2716, 2016.

[16] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S.-F. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 745–762.

[17] X. Wang, J. Ma, F. Li, X. Liu, Y. Miao, and R. H. Deng, "Enabling efficient spatial keyword queries on encrypted data with strong security guarantees," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4909–4923, 2021.

[18] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 697–708.

[19] Q. Tong, Y. Miao, J. Weng, X. Liu, K.-K. R. Choo, and R. Deng, "Verifiable fuzzy multi-keyword search over encrypted data with adaptive security," *IEEE Transactions on Knowledge and Data Engineering*, 2022, doi: 10.1109/TKDE.2022.3152033.

[20] R. Li, P. Wang, J. Zhu, J. Zhao, J. Di, X. Yang, and K. Ye, "Building fast and compact sketches for approximately multi-set multi-membership querying," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1077–1089.

[21] L. Xiao, O. Bastani, and I.-L. Yen, "An efficient homomorphic encryption protocol for multi-user systems," *Cryptology ePrint Archive*, 2012.

[22] T. Yang, J. Ma, Y. Miao, Y. Wang, X. Liu, K.-K. R. Choo, and B. Xiao, "Mu-teir: Traceable encrypted image retrieval in the multi-user setting," *IEEE Transactions on Services Computing*, 2022, doi: 10.1109/TSC.2022.3149962.

[23] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2401–2414, 2016.
[24] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *International Workshop on Public Key Cryptography*. Springer, 2013, pp. 1–13.
[25] X. Wang, J. Ma, X. Liu, and Y. Miao, "Search in my way: Practical outsourced image retrieval framework supporting unshared key," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2485–2493.
[26] Y. Li, J. Ma, Y. Miao, L. Liu, X. Liu, and K.-K. R. Choo, "Secure and verifiable multikey image search in cloud-assisted edge computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5348–5359, 2020.
[27] Y. Li, J. Ma, Y. Miao, X. Liu, and Q. Jiang, "Blockchain-based encrypted image storage and search in cloud computing," in *International Conference on Database Systems for Advanced Applications*. Springer, 2022, pp. 413–421.
[28] L. Song, Y. Miao, J. Weng, K.-K. R. Choo, X. Liu, and R. H. Deng, "Privacy-preserving threshold-based image retrieval in cloud-assisted internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13 598–13 611, 2022.
[29] K. W. Ahmed, M. M. Al Aziz, M. N. Sadat, D. Alhadidi, and N. Mohammed, "Nearest neighbour search over encrypted data using intel sgx," *Journal of Information Security and Applications*, vol. 54, p. 102579, 2020.
[30] H. Yan, Z. Chen, and C. Jia, "Ssir: Secure similarity image retrieval in iot," *Information Sciences*, vol. 479, pp. 153–163, 2019.
[31] M. Shen, G. Cheng, L. Zhu, X. Du, and J. Hu, "Content-based multi-source encrypted image retrieval in clouds with privacy preservation," *Future Generation Computer Systems*, vol. 109, pp. 621–632, 2020.
[32] Z. Xia, L. Jiang, D. Liu, L. Lu, and B. Jeon, "Boew: A content-based image retrieval scheme using bag-of-encrypted-words in cloud computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 202–214, 2022.
[33] Z. Xia, Q. Ji, Q. Gu, C. Yuan, and F. Xiao, "A format-compatible searchable encryption scheme for jpeg images using bag-of-words," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 18, no. 3, pp. 1–18, 2022.
[34] J. K. Mullin, "A second look at bloom filters," *Communications of the ACM*, vol. 26, no. 8, pp. 570–571, 1983.
[35] K. Christensen, A. Roginsky, and M. Jimeno, "A new analysis of the false positive rate of a bloom filter," *Information Processing Letters*, vol. 110, no. 21, pp. 944–949, 2010.
[36] C. Jin, C. Li, Z. Wang, Y. Zhang, and T. Zhang, "Sketch-based image retrieval with a novel bovw representation," in *International Conference on Multimedia Modeling*. Springer, 2016, pp. 621–631.
[37] J. M. Dos Santos, E. S. De Moura, A. S. Da Silva, and R. da Silva Torres, "Color and texture applied to a signature-based bag of visual words method for image retrieval," *Multimedia Tools and Applications*, vol. 76, no. 15, pp. 16 855–16 872, 2017.
[38] S. Guo, J. Xu, C. Zhang, C. Xu, and T. Xiang, "Imageproof: Enabling authentication for large-scale image retrieval," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1070–1081.
[39] E.-J. Goh, "Secure indexes," *Cryptology ePrint Archive*, 2003.
[40] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

Yingying Li received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 2017, and the Ph.D. degree from the School of Cyber Engineering, Xidian University, Xi'an, China, in 2022. She is currently a lecture with the School of Mathematics, Hangzhou Normal University. Her research interests include data security and applied cryptography.

Jianfeng Ma (M'16) received the Ph.D. degree in computer software and telecommunication engineering from Xidian University, Xi'an, China, in 1995. He was a Research Fellow with Nanyang Technological University, Singapore, from 1999 to 2001. He is currently a Professor and a Ph.D. Supervisor with the Department of Cyber Engineering, Xidian University. His current research interests include information and network security, wireless and mobile computing systems, and computer networks.

Yinbin Miao (M'18) received the Ph.D. degree from the Department of Telecommunication Engineering, Xidian University, Xi'an, China, in 2016. He was a Post-Doctoral researcher at Nanyang Technological University, Singapore from September 2018 to September 2019 and a Post-Doctoral researcher at the City University of Hong Kong from December 2019 to December 2021. He is currently an Associate Professor with the School of Cyber Engineering, Xidian University. His research interests include information security and applied cryptography.

Xiangyu Wang received the B.E. degree and Ph.D. degree from the School of Cyber Engineering, Xidian University, Xi'an, China, in 2017 and 2021. He is currently an Associate Professor with the School of Cyber Engineering, Xidian University. His research interests include data security and secure computation outsourcing.

Rongxing Lu (Fellow, IEEE) is a University Research Scholar, an associate professor at the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Canada. Before that, he worked as an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from April 2013 to August 2016. Dr. Lu worked as a Postdoctoral Fellow at the University of Waterloo from May 2012 to April 2013. He was awarded the most prestigious "Governor General's Gold Medal", when he received his PhD degree from the Department of Electrical & Computer Engineering, University of Waterloo, Canada, in 2012; and won the 8th IEEE Communications Society (ComSoc) Asia Pacific (AP) Outstanding Young Researcher Award, in 2013. Dr. Lu is an IEEE Fellow. His research interests include applied cryptography, privacy enhancing technologies, and IoT-Big Data security and privacy. He has published extensively in his areas of expertise (with H-index 81 from Google Scholar as of September 2022), and was the recipient of 9 best (student) paper awards from some reputable journals and conferences. Currently, Dr. Lu serves as the Chair of IEEE ComSoc CISTC (Communications and Information Security Technical Committee), and the founding Co-chair of IEEE TEMS Blockchain and Distributed Ledgers Technologies Technical Committee (BDLT-TC). Dr. Lu is the Winner of 2016-17 Excellence in Teaching Award, FCS, UNB.

Wei Zhang received the B.E. degree from the Department of Computer, Shandong Jiaotong University, Jinan, China, in 2021. He is currently pursuing the M.E. degree with the Department of Digital Information, Xidian University, Xi'an, China. His research interests include data retrieval and blockchain.