Hierarchical Bloom Filter Framework for Security, Space-efficiency, and Rapid Query Handling in Biometric Systems

Sumaiya Shomaji, Fatemeh Ganji, Damon Woodard, and Domenic Forte
University of Florida
{shomaji, fganji}@ufl.edu, and {dwoodard, dforte}@ece.ufl.edu

Abstract

Incorporating biometric technology into IoT could enable and assist a variety of futuristic applications such as cardless border crossing, secure access control in smart buildings, and patient tracking/anti-fraud in hospitals. However, existing approaches suffer from large storage requirements and latency in query handling. Additionally, there are privacy risks due to security breaches of raw biometric templates. In this paper, we propose a hierarchical bloom filter based identification system for large-scale biometric systems that reduces storage requirements while providing template security and rapid handling of queries. We address the challenge of incorporating a hash-based bloom filter with noisy biometric data by introducing a mathematical framework that is adaptive to characteristics of any biometric database. Our proposed architecture is implemented using a face database containing 30,000 facial templates and achieves 92.05% reduction in storage size with 99.82 reduction in average query time without sacrificing accuracy. Finally, the security is analyzed against potential tampering and collision attacks.

1. Introduction

Biometric technologies will allow Internet-of-things (IoT) systems to leverage an individual's unique biological properties for access control, to identify deviant behavior and individuals, etc. In large-scale IoT systems, biometric authentication shall require a big central database to store genuine templates. For instance, an entry point monitored by multiple cameras that facilitates access control among thousands would require a central database containing the face templates of employees. Further, each IoT camera would need to locally store and transmit this information for queries to the database. Such an approach suffers from three primary drawbacks. First, it could take a huge amount of space to store all the employee images as well as the targets seen by the cameras. Second, the search time to match each query from thousands of face data would encounter significant latency. Finally, template theft from the central database could be leveraged to perform various biometric template attacks [1]. In a naïve approach, encryption could be used to protect the stored templates from outsiders, but this adds additional decryption latency prior to any comparison of query data. Therefore, there is a need for authentication techniques that require less storage, offer fast processing of query, and protect encoded templates.

Bloom Filter (BF) can improve the space, query time, and privacy of such systems by introducing false positives. According to the theory of BF [7], to build a BF-based data structure containing n different elements with a template length of P, each element goes through a one-way hash algorithm k times, and the k different hash outputs represent that element in the BF. The BF can quickly verify that it contains a query by checking these k hashes even when the number of elements that it stores (n) is very large. In addition, since the BF only stores these k hash outputs for each entry instead of storing the raw data, the probability of recovering the original data is negligible. In other words, it also provides privacy to the data it contains. By setting $k \ll n$ and $k \ll P$, we posit that it is possible to design the BF-based biometric database that requires an insignificant amount of storage space compared to the original one. Due to small size, such BF-based database will also offer faster processing of queries. Finally, template security will also be achieved since during the BF design, a one-way hashing is performed on the raw templates. The main challenge with the legacy BF is that it cannot handle the fuzziness of data – an unavoidable characteristic of biometrics. To illustrate, let us assume that a BF-based biometric database contains templates from various people. Now, if a query template comes from a genuine person, but is noisy, the BF cannot find its match since the authentication is performed by checking indexes generated from multiple hash outputs of the query data. One bit of error in the query can result in a totally different hash output, thus, the index being checked becomes completely different and authentication fails. Therefore, even if a query from a genuine person is in BF, the system will never find a match for the query.

To overcome this issue, we propose a Hierarchical

Table 1. Comparison between Proposed Approach and Related Studies

Approaches	Refs	Biometric property	Accuracy	Time	Storage	Security	Use of Hash
Non-BF based	Nearest Neighbor [2]	fingerprint	medium	high	high	low	N/A
	Deep Learning [3]	any	high	high	high	low	N/A
BF based	BF for Iris [4, 5]	iris	medium	low	low	low	no
	BF for Face [6]	face	medium	low	low	low	no
	This paper	any (demonstrated on face)	high	low	low	high	yes

Bloom Filter (HBF) [8] based framework. HBF involves multiple levels, each with multiple BFs. During a template insertion into the HBF, it is segmented into multiple non-overlapping blocks and each block will be inserted into one BF in the hierarchy. The main idea is that noisy data should have at least some segments completely matching with its genuine template. If a pre-defined number of BFs from HBF can authenticate the noisy template, then the template is considered a member. This way, the architect can handle the fuzziness while utilizing the benefits of a legacy BF. We describe a complete HBF design methodology and mathematical framework that considers the amount of expected fuzziness and provides all design parameters for the HBF, including space requirement and number of matches needed for a desired false positive rate. We implement the proposed technique on a database containing 30,000 face templates and achieve 100% accurate authentication for large number of queries. The improvement in storage, query processing time, and security is also analyzed.

The remainder of the paper is organized as follows: Section 2 provides an overview of related work. Section 3 presents our proposed methodology, whereas Section 4 gives details on implementation, results, and application. Section 5 provides a discussion on the security. We conclude and describe future work in Section 6.

2. Related Work

There have been many studies trying to address biometric template security, storage efficiency, template query, and noise-tolerance. In Table 1, we have categorized them into two sets: non-BF based and BF-based. Non-BF-based approaches include machine learning, deep learning, indexing, nearest neighbour, etc. These methods have been explored for both identification and verification. Deep learning based techniques have been widely applied in noisy big data as they have shown promising results in learning variations within training samples. However, they require significant time to train and large storage to contain huge number of training samples [3]. Nearest neighbour search is a comparatively faster method which aims at fast searching for the nearest match. This requires less computation at a cost of reduced accuracy compared to deep learning based methods. While these approaches can be applied to any biometric modality, authors of [2] have implemented the nearest neighbour search method on fingerprint. For feature extraction they considered PCA and Linear Discriminant Analysis (LDA). In [9], the authors proposed an approach for fast querying using feature reduction through quantization. Raw face features were converted into Gabor-based features and then quantized to binary features. While this helped to reduce the overall size of the template and speed up the query processing, the feature conversion process was itself a time-consuming process due to the time needed for Gabor feature vector extraction [10]. Additionally, high-dimensional Gabor feature vectors could become very redundant [11]. Hence, a handful of studies attempt to address the issues associated with noisy big data. However, none of them provide storage and search efficiency for noisy data while enabling data privacy.

In the literature, the applicability of BF-like databases for providing security to the templates and faster operation has also been studied [4, 5, 6, 12, 13]. Authors have considered normalized iris templates in [4, 5] and local Gabor pattern based features for face templates in [6, 13]. Nevertheless, virtually all of those studies have not implemented BFs, as originally advised, i.e, by including the hash functions [7, 14]. To support why using hash functions is redundant in their approach, [13] discussed that their data is concise and collision-free. Nonetheless, use of hash functions in BFs has been advised to not only provide a concise representation of the data, but also to ensure the security due to the one-wayness of such functions. Excluding the hash functions indeed increases the vulnerability of databases to security threats [15]. Although non-cryptographic, hash-like transformations as used in [13] can be employed to reduce the probability of collision and ensure the conciseness, it is suggested to apply standard, cryptographic hash functions as an appropriate countermeasure against attacks targeting irreversibility of the data [15] (see Section 5). Thus, to our knowledge, in the context of biometric templates, our paper is the first work that follows the fundamental principles of BFs to address the storage-efficiency, fast query-processing, security and fuzziness issues altogether.

3. Proposed Methodology

First, we present a generic overview of the proposed system. The process mainly consists of two steps: (i) HBF-based database generation and (ii) Query processing for membership check in the generated database. During HBF-based database generation, a gallery set is considered for enrollment in the HBF architecture. As shown in Fig. 1, multiple pre-processing steps are performed on each image in the gallery set after which the processed templates go through quantization. Quantization is a mapping approach

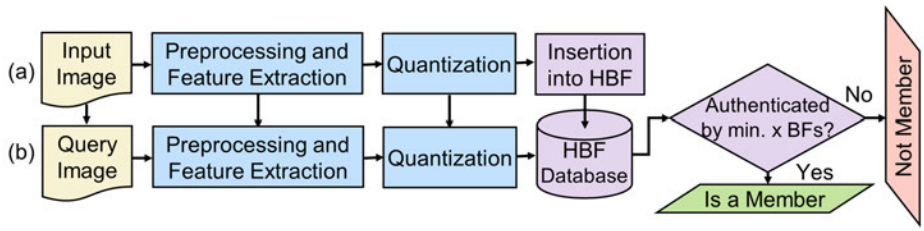

Figure 1. High level diagram of the proposed method: (a) Enrolling templates into HBF database and (b) Query handling.

to decrease the size of a template while reducing its noise. Finally, the quantized templates are inserted into the HBF. In the query processing step, the query image also goes through the same processing and quantization like gallery images. Then, the HBF is searched to see if it contains the query or a noisy version of it. If a query image gets matched by a minimum number of individual BFs, it is considered a member of that database. This threshold of minimum number of BFs is calculated at the HBF design stage based on some parameters discussed at the end of this section.

3.1. BF-Based Data Structures for Fast Query

A BF is a space-efficient, probabilistic data structure used to quickly check whether a data is definitely not in a set or possibly in it [14]. A BF is associated with certain parameters: the number of elements to be inserted in BF (n), false positive (FP) rate, the number of hash functions (k), and size of the BF (m) as listed in Table 2. This concise representation of a set of data can be achieved at the price of introducing FPs to the system. This means that when querying an unseen element, the BF can claim an element to be part of its set, although it has not been enrolled in the BF. However, the FP can be kept very low by configuring the design parameters of the BF.

Enrollment of an Element and Query Processing: As shown in Fig. 2, the working principle of a BF has two main processes, namely enrollment of an element (e.g., a biometric template) in BF, and query processing to test membership. For a given number of elements n, and the probability of a FP FP_{BF} (where $\text{FP}_{\text{BF}} \in [0, 1]$), one can determine the size of the BF (m) and the optimal number of hash functions (k) to maintain that desired FP_{BF}. These are the formulas used for determining m and k respectively:

Table 2. Notations and their descriptions.

Notation	Description
d	# layers in the HBF
e	# bits flipped by the noise in a template
k	# hash functions in a BF
l_i	# bits in a block of a template in i^{th} layer of the HBF
m	The length of BF in bits
n	# template
N	total # BFs in an HBF
p_{BF}	the probability of having a falsly-set-to-1 bit in a BF
p_t	the probability of flipping a bit location in a template
P	# bits in a template
w	# subsequent blocks concatenated in each layer of the HBF

$\text{FP}_{\text{BF}} \approx 0.6185^{m/n}$ and $k = (m \ln 2)/n$ [7]. The enrollment of a dataset (known as gallery set) into a BF starts with determining m and k for that BF. Once m is determined, a null vector of m bits is assigned as initial values in the BF. When inserting an element, i.e., a template, it is hashed k times first. The outputs of the hash functions denote locations or indices in the BF, where the template should be inserted. Since the outputs of the hash functions may exceed m, the modulus function is applied to them to obtain the location bounded within the range $[0, m]$. Afterward, the bits in the above-mentioned locations are set to "1" to complete insertion of the template. We illustrate this process with a toy example in Fig. 2 where a gallery set containing templates n_1, n_2, and n_3 are to be inserted in a BF with length m and one hash function is used for insertion. After employing the hash and the modulus on those raw templates, the outputs obtained are 2, 5, and 17 respectively. Now, these values will be considered as the index of the corresponding templates in the BF. The bits in the BF in those indexes are set to "1". During the query processing stage, the query templates (q_1 and q_2) are fed into the same hash and modulus functions for which the corresponding index values of 3 and 5 are obtained. The idea is, if these locations are already set to "1" in BF, the queries are considered as the members of the database. Hence, q_2 is a member of the database, whereas q_1 is not.

Hierarchical BFs: Hierarchical BFs (HBFs) have been introduced to address the problem of sub-string matching [8]. The core idea of HBFs is to check if a part of a string, i.e., an element, is inserted in the BF. To accomplish this, first, the input template (pixel intensities of grayscale images are considered as inputs) is normalized and quantized to get transformed as a binary string (details in in Section 4.1). Next, the string is splitted into fixed-size, non-overlapping blocks. Then, each block is independently inserted into standard, legacy BFs. This may result in an increase in FP due to combinations of sub-strings that are incorrectly reported as being in the BF [7].To tackle this problem, along with the sub-strings, their concatenations are also inserted in the filters, as illustrated in Fig. 3. As can be seen in this figure, the HBF is composed of N standard BFs with the same size m. When inserting a string, the smallest blocks of the string are added in the BFs that are present at the

Although the blocks may overlap, we consider non-overlapping ones.

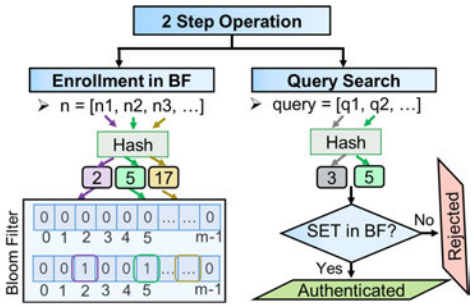

Figure 2. Steps taken to (a) enroll a new element in the BF, and (b) process the queries for membership search.

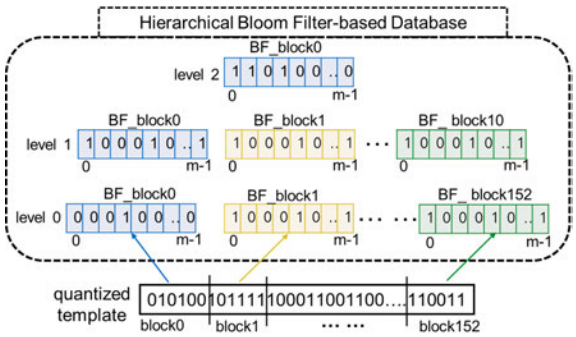

Figure 3. Schematic of an HBF. The templates are split into blocks. The smallest blocks are inserted in the BFs located at the lowest level, i.e., level 0. The whole string is inserted in the BF at the highest level d (here, $d = 2$).

lowest level in the HBF hierarchy. Consequently, the whole string is inserted in the BF at level d, i.e., the highest level. It has been demonstrated that the FP of such hierarchical structure can be drastically reduced, compared to the standard BFs involved in the HBF.

Performance Metrics for HBFs: Similar to standard BFs, the FP is a typical metric used to evaluate the performance of HBFs. In contrast to common scenarios, where BFs or HBFs are employed to store the data, in biometrics-related applications, we have to deal with noisy templates in both insertion and query processing events. Hence, it is evident that in addition to the FP rate, we should further define the following probabilities. The probability of *True Positive* (similarly, True Negative), TP (TN), refers to the cases, where an unseen template is correctly matched (incorrectly matched) to an any existing template in the HBF. More importantly, the probability of *False Negative* (FN) should be considered, when no match for a new query in the BF can be found due to fuzziness. As prime examples of the TP and the FN, the probability of matching, or finding no match for, an unseen noisy version of an existing template can be taken into account. Clearly, when such noisy template is inserted into/ queried from a BF, flipping even a single bit in a template results in having a wrong bit in a BF. Here the term "wrong" means that a bit is set to "1" due to the possible errors (collisions, the noise, etc.). Let p_{BF} denote the probability of such an event. Next, we aim to provide a better understanding of how p_{BF} is affected by template noise.

From the Noise in a Template to p_{BF}: Note that Table 2 provides a description of the terms used here. Consider a template, in which the noise has flipped e bits. Assuming that each and every bit can be noisy with an equal probability, we obtain $p_t = e/P$. When inserting such template in an HBF, the templates are divided into the blocks with length l_i ($0 \leq i \leq d$). Consequently, the probability of having no error in a block is $(1-p_t)^{l_i}$. Hence, when inserting a block in a BF, the probability of correctly setting a bit to "1" is $(1-p_t)^{l_i}/m$. If we assume that no collision between the hash functions could happen, the probability of having at least one erroneous bit after inserting a template is $\left(1-(1-p_t)^{l_i}/m\right)^k$. Suppose that no collision could occur during insertion of all n blocks (e.g., the first block of all n templates), the probability of having at least one error in the BF is $\left(1-(1-p_t)^{l_i}/m\right)^{kn}$. In fact, this probability is an approximation of p_{BF} that can happen in practice.

Defining the Thresholds for the Performance Metrics: As discussed above, our HBF is composed of standard BFs with an application defined FP_{BF}. When processing a block of a query, k bit positions in a BF are checked for a "1". Clearly, we have $\text{FP}_{\text{BF}} = (1-p_{BF})^k$. For a single BF, if we aim at defining a threshold, beyond which the performance of a BF is not acceptable we have $\text{FP}_{\text{TH}} = (1-p_{BF})^k \leq 0.6185^{m/n}$. Similarly one can compute the threshold of FN, TN, TP as presented in Table 3. Note that the desired level of these parameters are $\text{FP}_{\text{BF}} \leq \text{FP}_{\text{TH}}$, $\text{FN} \leq \text{FN}_{\text{TH}}$, $\text{TN} \geq \text{TN}_{\text{TH}}$, and $\text{TP} \geq \text{TP}_{\text{TH}}$. In order to define similar thresholds for an HBF, we begin with defining a desired level of FP that must be significantly lower than the FP of a legacy BF, i.e., $\text{FP}_{\text{HBF}} \ll \text{FP}_{\text{BF}}$. As demonstrated in [8], $\text{FP}_{\text{HBF}} \approx 0$, even if $\text{FP}_{\text{BF}} \approx 10\%$. Our goal is to define a threshold on the number of BFs (N_t), depending on k, p_{BF}, N, and l_i, so that if the number of BFs authenticating a template exceeds that, it is determined as "found" (see Table 3).

4. System Implementation and Results

4.1. Template Preparation

In this paper, we demonstrate the proposed HBF using facial templates. Pre-processing includes image normalization and quantization which are described below.

Image Normalization: For our experiments, four different datasets were used: Pinellas (database collected from the Pinellas County Sheriff's Office (PCSO)), Faces94 [16], FRGC [17], and Morph [18]. These datasets contain raw digital images of individuals. From these dataset, subsets were selected for the experiment. From Pinellas, 40,000 unique samples were selected for generating the mean vector. From rest of the datasets, only individuals having at least 4 noisy samples other than one genuine sample were

Next, we show how this assumption can be relaxed.

Table 3. FP, FN, TN, and TP thresholds for proposed HBF.

	Single BF	Proposed HBF
FP_{TH}	$(1-p_{BF})^k$	$\sum_{i=0}^{\lfloor N_t \rfloor} \binom{N}{i} (1-p_{BF})^{ki} (1-(1-p_{BF})^k)^{(N-i)}$
FN_{TH}	$1-(1-p_{BF})^k$	$\sum_{i=0}^{\lfloor N_t \rfloor} \binom{N}{i} (1-p_{BF})^{k(N-i)} (1-(1-p_{BF})^k)^i$
TN_{TH}	$1-(1-p_{BF})^k$	$\sum_{i=0}^{\lfloor N_t \rfloor} \binom{N}{i} (1-p_{BF})^{k(N-i)} (1-(1-p_{BF})^k)^i$
TP_{TH}	$(1-p_{BF})^k$	$\sum_{i=0}^{\lfloor N_t \rfloor} \binom{N}{i} (1-p_{BF})^{ki} (1-(1-p_{BF})^k)^{(N-i)}$

chosen. Therefore, 152 classes from Faces94, 302 from FRGC, and 557 classes from Morph were used. Since we considered four different datasets, the images in different datasets varied in terms of size, pose, illumination, etc. Therefore, a preprocessing step was performed on them to ensure identical standard before inserting them into the HBF. Similar to the flow presented in Fig. 1, each image went through the following major steps: (1) *Face detection*: After reading an input image, the face along with 68 facial landmark points were detected using dlib's pre-trained face detector [19]; (2) *Alignment*: Using the 68 landmark points, the faces were aligned by affine transformations; (3) *Masking*: A binary elliptical mask was created to perform logical AND operation of that mask with the aligned face image in order to keep only the important frontal parts of each face. This eliminated large forehead area, chin, neck, ear, and hair. After masking, we applied histogram equalization on the masked images to ensure uniform distribution of intensities. Note that there are numerous other ways to collect features, e.g., Principal Component Analysis (PCA), Kernel PCA, Convolutional Neural Network (CNN) or Local Binary Pattern (LBP), from masked images to boost the performance of the proposed system. However, since our main contribution is the application of the HBF, we have used the quantized features below for simplicity.

Quantization Process: To ensure a common dimension for all datasets, each image was resized to 70×70 after the masking step, resulting in 4900 features (pixel intensities) from each template. Next, we perform quantization. Features can be quantized in various ways (e.g., binarization, rounding, etc.). Here, we employ binary quantization, which has been used previously to compress features while speeding up the matching process [20]. For example, around 245 times faster matching speed was achieved in [21] through feature binarization. In [22], only 1% loss in accuracy was observed even after 16-24 times compression of feature data. We perform quantization in two steps. First, a mean feature vector is obtained from the feature values of a large population set. For this purpose, the Pinellas dataset was chosen over others for the mean vector calculation since it contained the highest number of classes (around one million). To calculate the mean vector, 40,000 individual samples were used. Second, any feature from a biometric template can be transformed into a binary representation using this mean. The features are compared to their associated mean values. Based on the feature value being higher/lower than the mean, the raw feature value will be replaced by 1/0.

Note that our gallery sets for the HBF included Pinellas, Faces94, FRGC, and Morph datasets while query sets only consisted of noisy samples from the latter three.

The main novelty of our paper is not in quantization, and the proposed framework is independent of the quantization method.

4.2. HBF Design and Implementation

As explained in Section 3.1, to design our HBF, we begin with defining desired levels of FP_{BF} and FP_{HBF}. Although it is suggested to have a low rate of the FP_{BF} , since we include multiple BFs in our HBF, we define a higher rate of the FP_{BF}, and at the same time $FP_{HBF} \approx 0$, similar to what has been devised in [8]. We set $FP_{BF} \approx 10\%$. In addition, we set $n = 30000$. Considering that $FP \approx 0.6185^{m/n}$ [7], for given n and a *desired* FP, one can compute m, e.g., in our setting $m \approx 140530$. The optimal k can be further computed as $k = (m \ln 2)/n$ [7], and in our setting it was $k \approx 4$. Additionally, for the HBF, we should define FP_{HBF} as $FP_{HBF} \ll FP_{BF}$. Moreover, let $p_{BF} \approx 0.5$ to account for the worst case scenario that is the probability of falsely setting a bit to 1 is approximately 50%. As k has already been computed for our desired FP_{BF}, we can approximate the total number of BFs (N) and N_t in a recursive manner (see Table 3). For our setting, we obtain $N \approx 165$ and $N_t \geq 20$ and we set $N_t = 21$.

As the next step in our design, to find the number of levels in the HBF and l_i, we should shift our focus to the number of bits that can be flipped due to the noisy nature of the templates. To set l_i, in an extreme scenario, we assume that the smallest blocks (i.e., the blocks inserted in the HBF at level 0) contain at least one bit flipped, with the probability close to one: $1-(1-e/P)^{l_0} \approx 1$. It is worth noting here that this probability can vary from one application to another and can be adjusted according to experimental results. When the values of e and P are known, we can compute l_0. According to our experimental results, on average $e/P \approx 16\%$. Setting $1-(1-e/P)^{l_0} \approx 0.995$, we obtain $l_0 \approx 32$. Note that in each level of the HBF, e.g., i^{th} level ($1 \leq i \leq d$), we concatenate w blocks from the level $i-1$, therefore, $l_i = w \cdot l_{i-1}$ ($1 \leq i \leq d$). Knowing l_i, our goal is to find the minimum value of d, and accordingly, the maximum value of w to fulfill the condition stated by the equation $N = \sum_{i=0}^{d} \lceil P/l_i \rceil$. Note that we aim to minimize d, and simultaneously, maximize w to improve space utilization. Solving the above equation in an iterative fashion, we obtain $w \approx 13$ and $d = 2$.

Hash function Used in Implementation: Hash functions can be classified as cryptographic and non-cryptographic. In the literature, it is suggested to employ a hash function from the first class, e.g., SHA256 can generate 256-bit outputs and offer an acceptable level of uniformity [23]. How-

ever, as reported in BF-related literature and observed by us in practice, this leads to an increase in the probability of collision after applying the modulus function. Therefore, we substitute Fowler-Noll-Vo (FNV) hash function for the modulus function [24]. This non-cyptographic hash function provides the advantage of not only reducing the collision probability but also limiting the output length to our desired length, i.e., m bits. By combining these two hash functions, we leverage their complementary advantages: security, fast processing, and low collision probability [25].

4.3. Results and Application Scope

Evaluation of the Performance of Proposed HBF: Once the HBF architecture is ready and gallery data are enrolled, we performed the query with 3,000 templates from Faces94, FRGC, and Morph. These 3,000 query templates can be categorized into 3 different types: (1) *Already Enrolled in HBF*: The first 1,000 query data are already enlisted in HBF which means HBF already has seen them. Thus, the HBF should trivially approve the membership of these individuals; (2) *Noisy template of Already Enrolled Person in HBF*: The next 1,000 samples are also known to HBF but HBF has not seen the exact same query template. In other words, the query is based on the same individuals' noisy templates (e.g., different expression or lighting). Ideally, the HBF should approve the membership of these individuals since they are members of the database; (3) *Never Enrolled in HBF*: The last 1,000 templates are from those individuals who were never enrolled in the HBF, so-called imposters, and should be obviously rejected.

After performing the query processing, the obtained result is depicted in Fig. 4. According to our proposed system, a query will be claimed to be a member of the database if it gets authenticated by at-least 21 individual BFs. From Fig. 4, we can see that the query templates in the first category are authenticated by all 165 BFs. For the second category queries, they are also templates from genuine people, but due to the presence of noise, many of the BFs fail to authenticate them. However, on average, 80 of the BFs in total can authenticate them. Therefore, they are claimed as members by HBF since $80 > 21$. Finally, the query templates from the last category are authenticated by less than 21 BFs, resulting in rejection by the HBF. Hence, none of the unseen data were authenticated by the proposed HBF.

Comparison of Accuracy Between Legacy BF and HBF: We also tested the same 3,000 queries for a legacy BF. We use the following metrics to evaluate the accuracy: (a) FPR: how many imposter values approved as a member out of the total number of imposter templates, (b) FNR: the number of enlisted genuine and noisy genuine templates rejected out of the number of all genuine templates, (c) TPR: the number of enlisted genuine and noisy genuine templates found as a member out of the number of all genuine templates, and (d) TNR: the number of imposter values rejected out of the number of imposter templates. Accord-

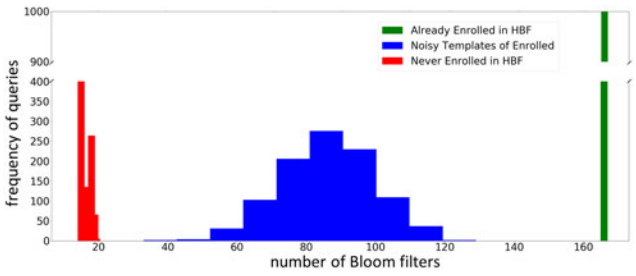

Figure 4. Histogram showing distribution of the total no. of BFs authenticating three types of queries: templates of already enrolled individuals in HBF, noisy templates of already enrolled individuals in HBF, and templates of never enrolled individuals in HBF.

ing to the theory of legacy BF [7], there should be no FN in BF. However, if we design a BF with genuine templates but query it with noisy templates, there are FNs as shown in Table 4. Increase in FPR for legacy BF also suggest that a lot of imposter templates got marked as genuine due the presence of noise. Using the proposed HBF, we eliminate these issue and achieved maximum possible detection accuracy across all metrics. To further compare our results with a recent similar work, the accuracy metric (FP, FN) of our approach have been compared to False Match Rate (FMR), False Non Match Rate (FNMR) of [6] because FP is comparable to FMR and so is FN to FNMR. As mentioned earlier, the authors of [6] designed a BF-based facial dataset and performed similar queries. They achieved FMR = 0% at the rate of FNMR = 20%. When they improved FNMR = 0%, they experienced FMR = 45%. However, we increased number of imposters up to 10,000 and performed query again. We still achieved FP = 0% with FN = 0% ensuring 100% accuracy. Moreover, using cryptographic hash (SHA256) in our approach assures better irreversability of templates than [6].

Time and Storage Comparison to Other Approaches: The 30,000 face templates occupied 37.5MB and 18.375MB for raw and quantized features respectively. As shown in Table 4, after enlisting this data in conventional BF, it required 0.0175MB. However, it introduced FN as observed in the above accuracy analysis. Compared to traditional BF, the proposed HBF required higher amount of storage which is still insignificant compared to the raw and quantized data. The HBF for 30,000 quantized templates resulted in 92.05% and 84.27% reduction compared to raw and quantized templates respectively. The comparatively smaller sized HBF also provides faster query processing. Table 4 shows that if a query is searched in an HBF

Table 4. Evaluation of the Performance of our HBF

Metric	Legacy BF	Proposed HBF
FPR	0.13	0
FNR	0.42	0
TPR	0.57	1
TNR	0.86	1

Table 5. Comparison between Various Approaches

Metric	Naıve Method		Legacy BF	Proposed
	Raw Templates	Quantized Templates		
Query Time	14.8s (average), 26.5s (worst case)	3.5s (average), 7.29s (worst case)	0.00058s	0.026s
Storage	37.5MB	18.375 MB	0.0175MB	2.89MB

Table 6. Complexity Analysis of Proposed System ($k \ll m, n, N$)

Metric	Naıve Method		Legacy BF	Proposed
	Raw Templates	Quantized Templates		
Query Time	$O(n)$	$O(n)$	$O(k)$	$N * O(k)$
Storage	$O(n)$	$O(n)$	$O(m)$	$N * O(m)$

database of 30,000 samples, it takes only 0.026s to get response on average whereas, the naïve based approach takes much longer time. If a query is compared to the samples inside a raw database, and the match is within the first half, it takes around 14.8s to get response, but if the query template is not in the database, matching is continued till the last template taking up to 26.5s to get response. Here, we used Euclidean distance to preform the comparison. Similar result is observed by the complexity analysis shown in Table 6. For a raw database, the time and storage requirement increase with the increase in no. of templates (n). However, in a legacy BF, the enrollment and searching complexity depends only on k and storage depends on m. Since we are using multiple BFs in HBF, the complexity further depends on no. of total BFs (N). Nonetheless, in the worst case scenario, the HBF provides better time and storage solution than the raw database as $k \ll m, n, N$.

Application Scope: The proposed HBF architecture is a perfect data structure for any database with issues during processing due to the large data volume and presence of noise. However, depending on the nature of the application, the proposed architecture might not be a standalone system. For example, if the purpose of a system is only tracking a human/object, then this HBF will work as a standalone system and answer only if that person is in that location's database or not. It will never answer, to which person in the database the query template belongs too, that means it will not do any kind of recognition. Therefore, if a system needs to do tracking as well as recognition, then the HBF can be paired with a traditional database comparison (e.g., cosine matching, Euclidean distance, Hamming distance, etc.). To illustrate this, let us assume a scenario. If multiple cameras are tracking people from multiple locations, and we need to recognize a query person, HBF will rapidly answer whether a person was seen in a specific camera location and its associated database. Then rather than brute force searching through all database locations, the BF isolates it to one. A detailed distance based search can then determine which gallery person the query template belongs to for one location rather than all of them.

5. Discussion on the Security of our Scheme

Despite the wide range of application of BFs in security-critical areas (e.g., forensic tools and intrusion detection systems), a natural question is whether applying BFs in biometrics-related scenarios can expand the attack surface of a system. The attacks relevant to our scenario include template recovery as well as injection of fake templates (so-called, tampering attacks). The former aim at recovering the biometric trait (i.e., the face in our case) from the information contained in the stored template, whereas in the latter an adversary injects the noisy biometric traits to authenticate non-registered persons. Here, we focus on how difficult it is for an adversary to *exploit* BF/HBF to launch these attacks. There are two inherent weaknesses of the legacy BFs that may help an adversary.

First and foremost, the adversary can benefit from the vulnerability of weak hash functions, mainly a high probability of collision, to retrieve information on a biometric trait. This type of attack has been well studied in the literature, and the countermeasures designed to stop them are commonly referred to as "irreversible" representation of the data. From the point of view of BFs, this type of attack can be categorized as finding a pre-image or a second pre-image for a BF [15]. In this regard, the probability of such events is $(1/2m)^k$. It has been demonstrated that this type of attack can be defended by choosing the BF parameters appropriately, namely, the optimal value of k and m. In our HBF scheme, in addition to employing these optimal parameters, we define a threshold on the FP and TN, which ensures that at least a number of BFs (N_t) in the HBF should authenticate several blocks of a template. This approach reduces the probability of finding a pre-image or a second pre-image to $(1/2m)^{N_t \cdot k}$. Our results confirm this by showing that the HBF does not authenticate an unseen template.

Secondly, as BFs are probabilistic data structures, an adversary can take advantage of this characteristic to cause worst-case behavior in terms of an increase in the FP or FN (similarly, a decrease in the TP and TN). In the context of tampering attacks, either a non-registered person could be falsely authenticated, or a registered one could not be authenticated. This problem can be addressed by adopting BF-based structures known to be more robust against slight changes in the elements inserted in the BFs. As a prime example of such structures, we adopt an HBF structure, whereby defining a threshold on the FP and TP, the success rate of such attacks is decreased considerably. Our experimental results for authenticating noisy templates and unseen templates support this claim. Specifically, the HBF data structure can clearly distinguish between a template, its noisy versions, and an unseen template.

6. Future Work and Conclusion

We have presented a secure, fast, and storage-efficient biometric authentication system using BF. By incorporating

hierarchy in traditional BF, we evade the issue of fuzziness in biometric templates. We presented a systematic HBF design methodology that is capable of determining design parameters for maximum possible noise and acceptable FP ratio. Even after designing the HBF assuming a maximum of 50% bit-flips in query data, we achieved 92.05% reduction in storage with 99.82% reduction in average query time. Future research will focus on investigating the best feature representations for the HBF templates, e.g., LBP, PCA, CNN, etc. By eliminating noisy features from the template, the performance of the HBF can be further improved. We will also investigate the storage savings and attack surface when FP and TP are further relaxed.

7. Acknowledgement

This project was supported in part by US Army Research Office grant under award W911NF-19-1-0102.

References

[1] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP J. on advances in Sig. processing*, vol. 2008, p. 113, 2008.

[2] H. binti Jaafar, N. binti Mukahar, and D. A. binti Ramli, "A methodology of nearest neighbor: Design and comparison of biometric image database," in *2016 IEEE Student Conf. on Research and Development*. IEEE, 2016, pp. 1–6.

[3] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514–525, 2014.

[4] C. Rathgeb, F. Breitinger, C. Busch, and H. Baier, "On app. of bfs to iris biometrics," *IET Biometrics*, vol. 3, no. 4, pp. 207–218, 2014.

[5] C. Rathgeb, F. Breitinger, H. Baier, and C. Busch, "Towards bf-based indexing of iris biometric data," in *2015 Intrl. Conf. on biometrics*. IEEE, 2015, pp. 422–429.

[6] M. Gomez-Barrero, C. Rathgeb, J. Galbally, J. Fierrez, and C. Busch, "Protected facial biometric templates based on local gabor patterns and adaptive bfs," in *2014 22nd Intrl. Conf. on Pattern Recog.* IEEE, 2014, pp. 4483–4488.

[7] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bfs for distributed systems," *IEEE Comm. Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.

[8] K. Shanmugasundaram, H. Brönnimann, and N. Memon, "Payload attribution via hierarchical bfs," in *Proc. of the 11th Conf. on Compt. and Comm. Sec.* ACM, 2004, pp. 31–41.

[9] T. A. Kevenaar, G. J. Schrijen, M. van der Veen, A. H. Akkermans, and F. Zuo, "Face recognition with renewable and privacy preserving binary templates," in *Fourth IEEE Wrks on Auto. Ident. Advanced Tech.s*. IEEE, 2005, pp. 21–26.

[10] S.-s. Liu and Y.-t. Tian, "Facial expression recognition method based on gabor wavelet features and fractional power polynomial kernel pca," in *Intrl. Symp. on Neural Net.s*. Springer, 2010, pp. 144–151.

[11] A. Vinay, V. S. Shekhar, K. B. Murthy, and S. Natarajan, "Face recognition using gabor wavelet features with pca and kpca-a comparative study," *Procedia Compt. Science*, vol. 57, pp. 650–659, 2015.

[12] P. Drozdowski, C. Rathgeb, and C. Busch, "Multi-iris indexing and retrieval: Fusion strategies for bf-based search structures," in *2017 IEEE Intrl. Joint Conf. on Biometrics*. IEEE, 2017, pp. 46–53.

[13] C. Rathgeb, M. Gomez-Barrero, C. Busch, J. Galbally, and J. Fierrez, "Towards cancelable multi-biometrics based on bfs: a case study on feature level fusion of face and iris," in *3rd Intrl. Wrks on biometrics and forensics*. IEEE, 2015, pp. 1–6.

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

[15] T. Gerbet, A. Kumar, and C. Lauradoux, "The power of evil choices in bfs," in *2015 45th Annual IEEE/IFIP Intrl. Conf. on Dependable Sys. and Net.s*. IEEE, 2015, pp. 101–112.

[16] L. Spacek, "Collection of facial images: Faces94," *Compt. Vision Science and Research Projects, University of Essex, United Kingdom, http://cswww. essex. ac. uk/mv/allfaces/faces94. html*, 2007.

[17] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *2005 IEEE Compt. society Conf. on Compt. vision and pattern Recog.*, vol. 1. IEEE, 2005, pp. 947–954.

[18] K. Ricanek and T. Tesafaye, "Morph: A longitudinal image database of normal adult age-progression," in *7th Intrl. Conf. on Automatic Face and Gesture Recog.* IEEE, 2006, pp. 341–345.

[19] D. E. King, "Dlib-ml: A machine learning toolkit," *J. of ML Research*, vol. 10, no. Jul, pp. 1755–1758, 2009.

[20] S. Petscharnig and K. Schöffmann, "Binary convolutional neural network features off-the-shelf for image to video linking in endoscopic multimedia databases," *Multimedia Tools and App.s*, vol. 77, no. 21, pp. 28 817–28 842, 2018.

[21] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Fusion and binarization of cnn features for robust topological localization across seasons," in *2016 Intrl. Conf. on Intelligent Robots and Sys.* IEEE/RSJ, 2016, pp. 4656–4663.

[22] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[23] D. Rachmawati, J. Tarigan, and A. Ginting, "A comparative study of message digest 5 (md5) and sha256 algorithm," in *J. of Physics: Conf. Series*, vol. 978, no. 1. IOP Publishing, 2018.

[24] G. Fowler, L. C. Noll, and K.-P. Vo, "Fowler/noll/vo (fnv) hash," http://www.isthe.com/chongo/tech/comp/fnv/ [accessed 9 April 2019].

[25] L. Chi and X. Zhu, "Hashing techniques: A survey and taxonomy," *ACM Computing Surveys*, vol. 50, no. 1, p. 11, 2017.