Secure fingerprint hashes using subsets of local structures

Tom Effland[a]*, Mariel Schneggenburger[a]*, Jim Schuler[a]*, Bingsheng Zhang[ab], Jesse Hartloff[a], Jimmy Dobler[a], Sergey Tulyakov[a], Atri Rudra[a], Venu Govindaraju[a]

[a]University at Buffalo, SUNY, Buffalo, NY, USA
[b]National and Kapodistrian University of Athens, Athens, Greece

ABSTRACT

In order to fulfill the potential of fingerprint templates as the basis for authentication schemes, one needs to design a hash function for fingerprints that achieves acceptable matching accuracy and simultaneously has provable security guarantees, especially for parameter regimes that are needed to match fingerprints in practice. While existing matching algorithms can achieve impressive matching accuracy, they have no security guarantees. On the other hand, provable secure hash functions have bad matching accuracy and/or do not guarantee security when parameters are set to practical values.

In this work, we present a secure hash function that has the best known tradeoff between security guarantees and matching accuracy. At a high level, our hash function is simple: we apply an off-the shelf hash function on certain collections of minutia points (in particular, triplets of minutia "triangles"). However, to realize the potential of this scheme, we have to overcome certain theoretical and practical hurdles. In addition to the novel idea of combining clustering ideas from matching algorithms with ideas from the provable security of hash functions, we also apply an intermediate translation-invariant but rotation-variant map to the minutia points before applying the hash function. This latter idea helps improve the tradeoff between matching accuracy and matching efficiency.

Keywords: security, fingerprints, biometrics, cryptography

1. INTRODUCTION

Biometrics in general, and fingerprints in particular, have the potential to replace text-based passwords as the primary mode of authentication. Text based passwords are prevalent because (i) there exist secure hash functions for strings that are computationally hard to invert and (ii) matching of hashes can be done efficiently since the hash values are themselves strings. For fingerprints, there has been work on both axes of security[1–4] and of matching.[5] However, to date, there is no work on hashes on fingerprints that has both provable security and practical matching accuracy. Following our recent work,[6,7] we continue the search for hashes on fingerprints that have a better tradeoff between provable security and achievable practical matching accuracy.

Existing work on provably secure hash functions treats the fingerprint as sets. This is the minimum requirement for a hash function to work on fingerprints; due to the way scanners read fingerprints, it is impossible to represent the fingerprint as a vector. Such a hash function h works as follows: Given two fingerprints f_1 and f_2 (as sets), the fingerprint f_2 is matched to $h(f_1)$ *if* $|f_1 \cap f_2|$ is sufficiently large. In other words, two fingerprints are matched if they have a large set intersection. Further, these schemes would work for any sets irrespective of whether or not they came from fingerprints. As a specific example, consider the following two cases where we have $|f_1 \cap f_2| = 3$. In one case the three common points are immediately adjacent, while in the other case the three points are far apart. These two situations are different (in the former case, intuitively, the match is much stronger), but existing secure hash functions treat both cases as the same. This results in poor matching accuracy. Another issue with most such secure hash functions is that, even though these schemes have provable security, the security only works at parameter regimes that are not useful in practice.[6]

On the other hand, existing matching algorithms heavily utilize the geometry of the fingerprint. So for example, in the above scenario, matching algorithms will give more "weight" to the case when matched points

*These authors contributed equally to this work and are listed alphabetically. Send correspondence via email to jcschule@buffalo.edu.

are close to each other. Unfortunately, these matching algorithms need the fingerprints to be stored in the clear, which implies absence of any security. This leads to the following motivating question for our work:

> *Is it possible to design secure hash functions that exploit the geometry of fingerprints for better matching accuracy as well as have provable security for parameter regimes that are needed in practice?*

In this work, we answer the question in the affirmative. In particular, our scheme uses ideas from matching algorithms along with ideas for secure hashes on strings. In a nutshell, here is how our scheme works. Given a set of n minutia points, for each point, we consider the two closest points to form a total of n triplets. Then, we convert the length of the sides of the corresponding triangle (as well as some angular information) into a fixed length string in a deterministic way (let us call this the *canonical string* of the triangle). During the matching phase, we build n triplets as above for the candidate fingerprint and construct the canonical string for each of these triangles. Finally, we try to match the hashes of as many canonical strings as possible with the existing hash values. In hindsight perhaps our scheme looks very simple, since we use the idea of matching triplets (which is common in existing matching algorithms[8–10]) and simply apply an off-the-shelf string hash function on such triangles. However, to make this idea work we need to address issues *both* in security and matching.

Any matching algorithm has to deal with the fact that the candidate fingerprint (even if it is a match for the enrolled fingerprint) can be rotated and translated differently from the enrolled fingerprint. There are three ways to deal with this issue. The first is to add extra alignment information to the enrolled fingerprint to assist in the matching process later on. However, we cannot store this information in the clear since this extra information will reduce the security of the scheme (and it is much harder to quantify this loss in security). In earlier work,[6] we used this extra information when we tried to match fingerprints using the fuzzy vault hash.[11] In particular, this extra information was used to get an estimate on the limit of the best possible matching numbers, though we *ignored* the security leakage from using this extra information. To complement the existing result, in this paper we ran experiments on the fuzzy vault scheme without the extra alignment information.

The second option is to first use a translation- and rotation-invariant hash on the minutia points, and then use an existing, provably-secure hash function such as fuzzy vault[11] or Pinsketch[12] on the intermediate set of points. (Both of these schemes do not handle translation and rotation on their own). In our previous work,[7] we used fuzzy vault on top of a translation- and rotation-invariant map based on paths of minutia points. However, the matching numbers for these schemes are inferior to the results in this paper. Informally, the reason for inferior matching numbers is that these hashes lose a lot of the global information about the fingerprints. In particular, this map does not distinguish between the case when the entire fingerprint is translated and rotated versus a situation where two parts of the fingerprint are translated and rotated independently. (The latter would not happen in real life, of course, but this increases the chances of an impostor fingerprint getting matched.)

The third option is to brute-force through all translation and rotations (in some quantized way) on the candidate fingerprint and then use the existing matching algorithms of fuzzy vault or Pinsketch. The advantage of this scheme (over the previous option) is that in this case the matching does use consistent global information about the fingerprint. However, this scheme is slow and hence undesirable in practice.

In this work, we combine the second and the third options. We store the relative distance between points in triplets of triangles, which are translation invariant. However, we store the angles as-is. That is, this scheme is translation invariant but *not* rotation invariant. While matching, we consider many possibilities of rotation values. This cuts down on search space for translations while still allowing us to have some success in in deterring impostor prints from having a high overlap. The final piece is to store the hashes for not just the enrolled fingerprint but as for the set of rotated fingerprints, where the rotations come from a pre-determined set. (This increases the space requirement but significantly increases the matching accuracy.)

It turns out that the scheme as stated above is *not* secure. In particular, a simple brute-force algorithm suffices to break the above scheme since, for our parameter settings (which seem to be needed to get good matching), one can just enumerate all possibilities for the triangles. To get around this issue we consider *triplets* of triangles instead of the triangles themselves. This increases the search space and rules out the naïve brute-force search attack. In Section 4, we provide a formal security analysis of our scheme. In particular, we prove the security

of our scheme under the assumption of honest-but-curious parties and under the random oracle model (i.e., we assume that the SHA-256 function behaves as a completely random function).

Finally, we would like to point out that, once we have the intermediate collection of triplets of triangles, nothing stops us from using a hash function other than SHA-256. In particular, it is natural to wonder if one can use the fuzzy vault. For comparison, we also implement our scheme with the fuzzy vault, which achieves identical matching numbers as our scheme above with two significant disadvantages. First, there is no security guarantee for this scheme. Second, if we want to increase the security, say in the sense of our earlier paper,[6] then the space requirement for the hash function is huge. In particular, for each fingerprint (whose size is about half a kilobyte), each hash value would be of the order of several *gigabytes*!

2. RELATED WORK

The majority of the algorithms proposed for creating privacy preserving fingerprint templates operate with minutia based fingerprint templates. The challenges in matching sets of minutia corresponding to two fingerprints include their variable size and coordinate changes due to skin deformation. In order to deal with these challenges, a common strategy of minutia subset matching is employed: subsets containing a few neighboring minutia are found, the candidate pairings of such subsets in two fingerprints is performed in the first step of matching procedure, and the pairs of matched subsets are grouped for the final match result in the second step. Such strategy assumes that the matching of local neighborhoods is readily approximated by the affine transformation, whereas the matching of whole fingerprints is not. The minutia subset matching strategy has been used for traditional fingerprint matching,[13] as well as, for fingerprint indexing.[14] The goal of the current paper is to construct privacy preserving fingerprint templates utilizing a similar minutia subset based approach to matching.

Possibly one of the most studied systems for template security in fingerprints is the fuzzy vault. Nandakumar et al.[4] present a fuzzy vault implementation with probably the best known accuracy. To help with alignment, the authors store a "hint" in a high-curvature datum in the locked vault to assist with alignment. The security of the fuzzy vault can be described using pre-defined attacks and entropy measures.[15] This system does perform better than ours in terms of matching accuracy, though it is uncertain if storing the high-curvature datum in the clear affects the security of the system. In addition they, along with much of the work on fuzzy vaults, present numbers for specific sizes of the secret polynomial. These values are chosen at points where the system has reasonable security, though at other values the system may not be secure at all. Our proposed system is much more versatile and is secure at a wide range of parameters ranging over the entire ROC curve.

Nagar et al.[16] showed that the matching accuracy of the fuzzy vault system can be improved without sacrificing security by utilizing minutia descriptors. These descriptors capture data from the area surrounding each minutia point. This data is not stored in the vault, yet it used for locking and unlocking the vault. Since it is not stored, an attacker presumably must guess all the bits of this data so the entropy of the vault will be increased by the number of bits used. They show that the vault will have a reasonable amount of entropy in the average case with their setup.

Each single minutia in the fingerprint template corresponds to a point in the fuzzy vault in traditional fuzzy vault implementations.[4,16] Consequently, the matching of fuzzy vault is analogous to full minutia set matching and does not involve local minutia subsets. To bypass the inherent limitation of such a matching approach, the algorithms incorporating local minutia subsets in the fuzzy vault have been proposed as well. Hartloff et al.[7] used local configurations of 5 minutia positions to extract rotation and translation invariant features, which were concatenated to form a point for fuzzy field encoding. In addition to eliminating the need for storing alignment information needed for traditional fuzzy vaults, this method also utilized a subset matching strategy - the matching of local minutia subsets is equivalent to matching corresponding points in the fuzzy vault. Due to using rather large minutia subsets and the ability to match only a small number of them in two fingerprints, the presented method needed to use small degree encoding polynomials and, as consequence, a large number of chaff points to preserve the security properties. In addition, this work used multiple fingerprint reading during the enrollment phase which increases the matching accuracy.

The privacy preserving methods incorporating the subset matching strategy and not involving fuzzy vaults have been proposed as well. Kumar et al.[17] perform non-invertible transformation of local minutia k-plets ($k =$

3, 4) by means of symmetric polynomial functions. The resulting hashes from two fingerprints are subsequently used to calculate global transformation parameters (rotation and translation), as well as the confidence of the match between two fingerprints. Ferrara et al.[2] converted minutia neighborhoods of pre-determined radius to the feature vectors of fixed dimension (minutia cylinder code, or MCC, representation), and subsequently performed randomized projection of such feature vectors to get the non-invertible hashes. Such local hashes can be matched in two fingerprints, and the global matching score is calculated as a combination of matching scores between local hashes. Due to loss of information during the non-invertible transformations of local neighborhoods, both these methods report decreased matching performance relative to original matching algorithms.

In the algorithm proposed in the current paper we investigate the minutia subset matching strategy. In contrast to the above referenced non-fuzzy vault methods, we construct non-invertible transformations using cryptographic hashes with well defined security properties. Also, we are not constructing hashes of individual neighborhoods, but of their combinations. Although our algorithm is similar to the fuzzy vault method utilizing translation invariant features of minutia neighborhoods,[7] we avoid storing non-encrypted geometry information in the database (thus avoiding database cross-reference attack vulnerability of fuzzy vaults[18]) and require significantly less memory per template due to not storing chaff point information.

3. SYSTEM OVERVIEW

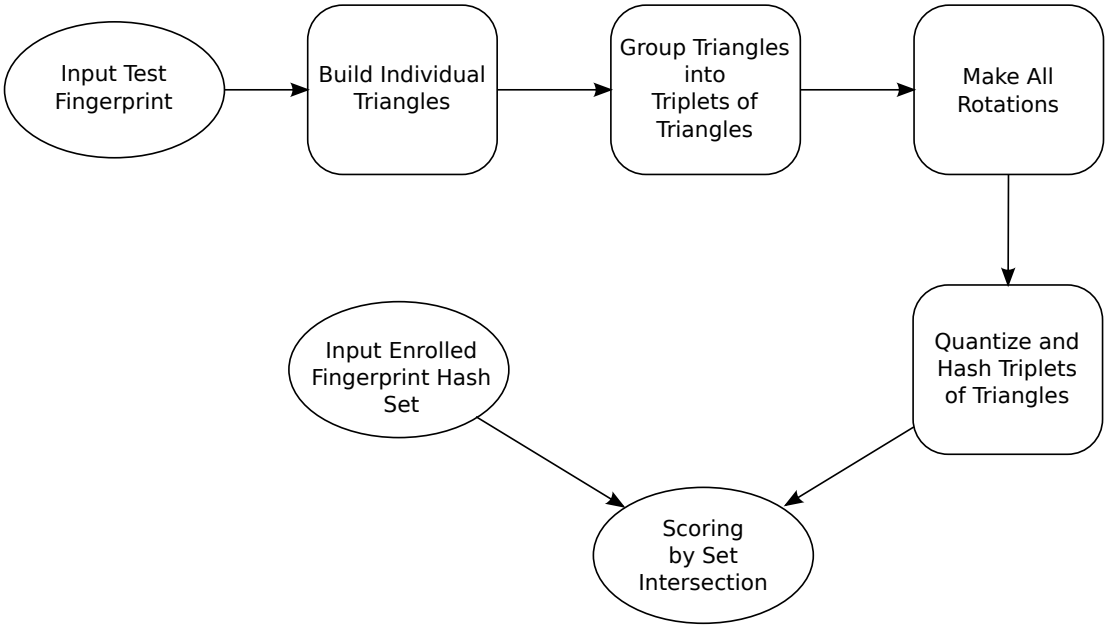


Figure 1. Overview of System Process

Figure 1 presents an overview of our matching algorithm. In the rest of the section, we will elaborate on each of the tasks. Specifically, we describe first how we form triplets from individual minutia points of a fingerprint, which we will call a triangle. We then combine three of these triangles into what we call a triplet of triangles. Next, we describe the manner in which we consider all rotations of the triplets, which we then quantize. Finally, we describe how to produce a match score using standard hash functions. This scheme is invariant in translation, but variant, by design, in rotation.

The hash scheme for the enrollment fingerprint is the same as the one described below without the scoring function. (That is, in Figure 1, we will do everything till the part where we have to compute a score.) In

particular, note that this implies that the fingerprint is stored as a set of hashes of quantized triplets of triangle overs the set of pre-determined rotations.

A fingerprint is represented as a set of n minutia points, or major features. These major features are, for example, ridge endings, ridge bifurcations, or islands. A minutia point consists of an x value, y value, and a θ value, where θ is the direction of the ridge at which a minutia point is located.

3.1 Formation of Triangles and Triplets of Triangles and Rotation

For a fingerprint we form triangles, a group of three minutia points, from each of the n minutia. We do this by considering a minutia point and the next two closest minutia points. For security, we use triplets of these triangles. If we were to use all of them, there would be $\binom{n}{3}$ triplets of the n triangles. However, we only consider those triplets whose triangles are within a given distance threshold from each other. In order to call a fingerprint a match with the enrolled fingerprint, a certain number of these triplets must match between the two fingerprints, which is another threshold we will discuss.

Build Individual Triangles: First, we construct the individual triangles from the set of minutia points. Each minutia point is considered individually and is converted into a triangle with its two closest neighbors. The two closest neighbors are those with the smallest Euclidean distance, in x and y, to the minutia point in consideration. With these three points, the original plus the two closest, we will construct a triangle. We label the three points p_0, p_1, and p_2, which we will order in a deterministic manner as follows. The minutia point with the smallest x value is labeled p_0. Ties in the value of x are broken consistently, with the point with the larger y chosen to be p_0. Of the two remaining points, that with the larger y value is chosen to be p_1, and the remaining point is p_2. In the case of ties between y values when distinguishing p_1 and p_2, the algorithm chooses the point with the smaller x value to be p_1. Next, we translate all three points such that p_0 is located at $(0, 0)$ by subtracting the x and y values of p_0 from all three minutia points. This maintains the distance between the points, but allows us to compare triangles more easily. This distance property creates triangles that are invariant in translation, but variant in rotation. The rotation variance exists because we did not consider the θ values of the minutia points in triangle construction (as well as the fact that the x and y values of the points p_1 and p_2 change with rotation). An example of the construction of an individual triangle is seen in Figure 2.

For each individual triangle we then calculate the center point, an important and descriptive piece of data that will be used in the grouping of triangles. We call the center of the triangle to be the average x value and the average y value of the three minutia points.

This returns one triangle, t_i, $0 \leq i \leq n$, per minutia point, so there does exist some overlap of triangles in their entirety. This overlap will be considered carefully when grouping triangles during the building of triplets in order to prevent certain attacks from compromising our system.

Group Triangles Into Triplets of Triangles: After the set of triangles has been constructed for each fingerprint, we group these triangles into sets of size three in a deterministic manner as follows. We will change the representation of a fingerprint from a set of triangles, $\{t_0, ..., t_n\}$ to a set of triplets of triangles $\{T_0, ..., T_N\}$. Given n constructed triangles, we consider $\binom{n}{3}$ triplets by forming sets of all possible combinations of three triangles. When forming all $\binom{n}{3}$ combinations, if two of the triangles are farther apart based on the Euclidean distance between their respective centers than some given distance threshold (in practice this is 100), we choose to ignore the triplet that would be formed by these triangles. This saves on computation time, yet does not negatively impact matching later.

Similarly, if a triplet of triangles is composed of less than a certain number of minutia points v, we will disregard the triplet and not use it in any matching. As described in Section 4, there exists a possibility of a triplet of triangles constructed with only $4, 5$, or 6 distinct minutia points. A triplet of this type can be used to compromise the security of the system. Therefore, we discard triplets composed of 6 points or less. However, this impacts the matching accuracy, so we have included the results of tests done using various thresholds for the minimum number of points in each triplet of triangles. This is illustrated in more detail in Section 5.

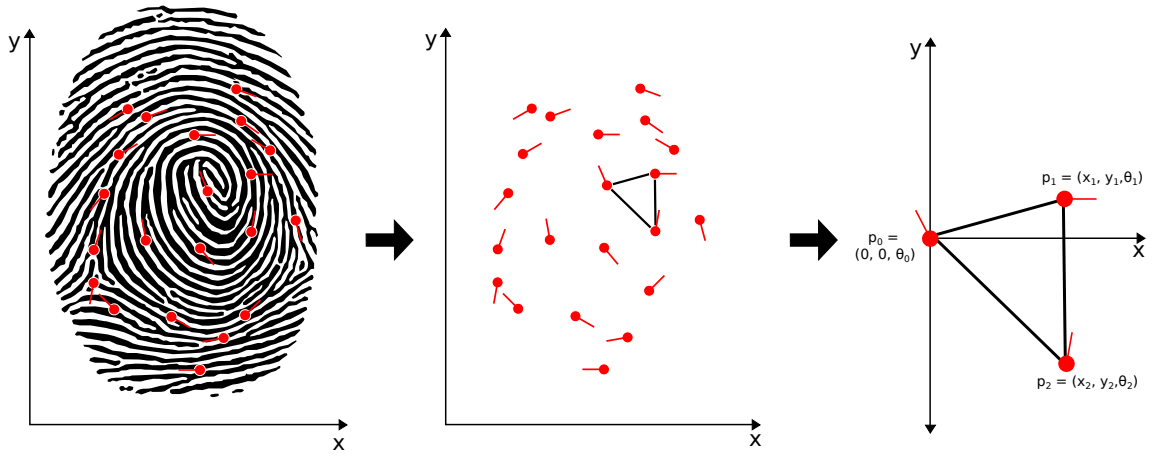


Figure 2. The first arrow shows a sample formation of a triangle from a minutia and its two nearest neighboring minutia. The second arrow shows that the triangle is translation invariant with the leftmost minutia at the origin, but is not rotation invariant.

Each fingerprint is now uniquely represented by a set of triplets of triangles, $\{T_0...,T_N\}$, where each T_j is a set of three triangles, $\{t_{j_0}, t_{j_1}, t_{j_2}\}$.

Make all Rotations: It is likely, however, that different fingerprint readings will be taken at different angles. We perform a series of rotations to assist in the matching process. This accounts for the fact that the triangles are not rotation invariant, even though they are translation invariant.

In each triangle, the rotation angle is added to the θ values associated with p_0, p_1, p_2, and the rotated x and y coordinates of these points are found using trigonometric identities. After each triplet is rotated using this method, a new rotated fingerprint is formed. Note that we reapply the method above for determining the ordering of the points p_0, p_1, and p_2, since the order of the points could change after rotation.

We rotate the fingerprint from $-50°$ to $50°$ by intervals of $2°$, so the total number of rotations is $d = 101$. After we finish all of the rotations on the fingerprint, we take the triplets found for each rotation and quantize, concatenate, and score the test fingerprint as described below.

3.2 Quantize and Hash Triplets of Triangles

Once we have a representation of each fingerprint as a set of triplets of triangles rotated at each rotation, $\{T_0, ..., T_k\}$, where $k \leq N \cdot d$,* we quantize these triplets into elements of $\mathbb{F}_{2^q}$ (where q is determined by the binning parameters) as follows:

Binning: First, we quantize the individual triangles of the set by quantizing its individual elements. We take a triangle $t = (x_1, x_2, y_1, y_2, \theta_0, \theta_1, \theta_2)$ as in Figure 2 and quantize each component using three binning functions, one each for x, y, and θ. We note that using fixed-width bins does not achieve uniformity in the quantized space, so we use variable-width bin thresholds to get close to uniformity. These bin thresholds are determined experimentally by finding uniform bin volumes and outputting variable-width bin thresholds that achieve these fixed volumes. These thresholds are found by fitting the desired number of bins to the

*We note that in practice, k is usually much smaller than this, since we do not allow duplicate triangle triplets in the set. This is because it takes up considerably more space while not improving matching numbers. It also does not impact our security since an adversary would not need to guess any particular triangle triplet more than once. See Section 4 for more details.

data from the first database (DB1) from the 2002 Fingerprint Verification Competition (FVC2002). The thresholds are then tested with the other databases from the same year. This near-uniformity is necessary to maximize the entropy of the quantized triplets; a necessary condition for security.

Quantizing a Triangle: Each x value is quantized to an element of $\mathbb{F}_{2^p}$ (where $p = \lceil \log_2(\text{Number of } x \text{ bins}) \rceil$). Each y is quantized to an element of $\mathbb{F}_{2^r}$ (where $r = \lceil \log_2(\text{Number of } y \text{ bins}) \rceil$). Each θ is quantized to an element of $\mathbb{F}_{2^s}$ (where $s = \lceil \log_2(\text{Number of } \theta \text{ bins}) \rceil$). After each value is quantized, the triangle t is formed by concatenating these values in the order listed above. So we have each triangle $t = x_1 \circ x_2 \circ y_1 \circ y_2 \circ \theta_0 \circ \theta_1 \circ \theta_2$. The x values will always be positive values, whereas a y value can cover a wider range of values since it can be either positive or negative.

Quantizing the Triplet: Next we concatenate the three triangles in a triplet by a deterministic ordering as follows. With each triangle, we store its original (non-translated) center point in the fingerprint $t_{\text{center}} = (x_{\text{center}}, y_{\text{center}})$. The average of the three center points of the triplet is taken to find the center point of triplet $T_{\text{center}} = (X_{\text{center}}, Y_{\text{center}})$. We then order the triangles by closest Euclidean distance from the triangle center to the center of the triplet,

$$\text{dist}(t_{\text{center}}, T_{\text{center}}) = ||t_{\text{center}} - T_{\text{center}}||_2$$

and concatenate the three triangles in this order, yielding our final quantized value

$$Q = t_1 \circ t_2 \circ t_3$$

where

$$\text{dist}(t_{1_{\text{center}}}, T_{\text{center}}) \leq \text{dist}(t_{2_{\text{center}}}, T_{\text{center}}) \leq \text{dist}(t_{3_{\text{center}}}, T_{\text{center}})$$

In the event of a distance tie, we break the tie arbitrarily by declaring the triangle with the leftmost center to be closer. After this quantization process, we now have each fingerprint represented as a set of values Q where $Q \in \mathbb{F}_{2^q}$, where $q = 6 \cdot p + 6 \cdot r + 9 \cdot s$. We multiply p and r by 6 because within a triplet of triangle there are 6 x values and 6 y values. We multiply s by 9 for a similar reason; within a triplet of triangles we consider 9 θ values, three from each triangle. We then append the user's publicly stored 256-bit salt to the end of this value and hash the entire result using SHA-256. This gives us our $h(Q)$ value. In theory any provably secure hash function will work for this purpose. This is for security and is discussed in detail in Section 4. We also have timing tests for this process in Section 5.

3.3 Scoring

Each fingerprint is uniquely represented as a set of triplets of triangles, now quantized and hashed to $h(Q)$ values. We find the level of matching between prints via a scoring, determined by the set intersection of the hashed quantized values.

Scoring by Set Intersection: To score the fingerprints, a basic set intersection method is used. The set of $h(Q)$ values that were found in the quantization and hashing process are now compared across two fingerprints, f_1 and f_2. If $h(Q_1) = h(Q_2)$ for some Q_1 in f_1 and Q_2 in f_2, it is considered a match and the intersection score is increased by one. Thus, for two fingerprints to have a score greater than 0, there must be at least one matching triplet of triangles between them after the hash is implemented. We consider two fingerprints to be a match when the set intersection score is greater than or equal to a given threshold. Using the binning parameters from above, we experimentally determine this threshold to be one.

3.4 Fuzzy Vault

For comparison purposes, we also generate matching numbers given by the fuzzy vault system.[11] To do this, the fingerprints are enrolled using the same quantization as in the triplets of triangles scheme, with 100,000 chaff points per fingerprint.

The Fuzzy Vault system first generates a polynomial p of degree z, then calculates $p(Q)$ for each quantized triplet of triangles Q. These values are recorded, along with random chaff values. To unlock the vault, one

first takes each triplet of triangles Q' in the fingerprint attempting to unlock the vault and determines finds if a corresponding point for that value exists in the vault. The unlocking algorithm then takes all points found this way and attempts to use Reed-Solomon decoding techniques to find the interpolating polynomial. If enough genuine points were found, then the output of the above will be the original polynomial.

The fuzzy vault security parameter was defined by Hartloff et al.[6] to be

$$\lambda \stackrel{\text{def}}{=} \sqrt{cz} - g, \tag{1}$$

where z is the degree of the polynomial, c is the number of chaff points, and g is the number of genuine points.

4. SECURITY

Here, we formally analyze the security of our proposed scheme. We first define the model of adversary we are working under. We also mention how we overcome the multiple authentication server vulnerability that hinders the fuzzy vault scheme.

4.1 Adversary models

Our security setting and adversary model are similar to the ones considered in the conventional fuzzy-vault-based authentication schemes.[4,16] The security threats faced by our proposed biometrics authentication scheme primarily come from the employment of untrusted (third-party) servers for our authentication service. In practice, those servers could either be naturally malicious or compromised by some external adversaries. We will examine the security of our scheme against so-called honest-but-curious servers; that is, servers that always follow their protocol specification but try to silently recover the users' fingerprint templates.

Hence, the system security is defined in terms of the users' privacy — whether or not the enrolled fingerprint templates are kept confidential. Our analysis is conducted in the random oracle model, where SHA-256 is viewed as an ideal random function that responds to every unique query with a uniformly random response expect that it responds with the same hash digest to the same query. Assuming the triplets of triangles are randomly distributed, we give the following security definition:

DEFINITION 4.1. *Let $k \in \mathbb{N}$ be the cardinality of the triangle triplet set. We say a single user's fingerprint template is (τ, ε)-private if for all non-uniform τ-time adversaries $\mathcal{A}$, the winning probability in the game where the challenger samples k random triangle triplets $(Q_1, \ldots, Q_k) \leftarrow (\mathbb{F}_{2^q})^k$ and a salt* $\text{salt} \leftarrow \{0,1\}^{256}$ *is*

$$P_{win} := \Pr\left[\mathcal{A}(H(Q_1 \circ \text{salt}), \ldots, H(Q_k \circ \text{salt})) \in \{Q_1, \ldots, Q_k\}\right] \leq \varepsilon \ .$$

The output of the adversary $\mathcal{A}$ is defined as an element of $\mathbb{F}_{2^q}$. Namely, given the emulated hash digests, we consider that the adversary wins if he or she is able to guess one of the triplets of triangles correctly.

4.2 Security analysis

Suppose SHA-256 is collision-resistant. We consider the hash digests of distinct values in our scheme to be collision-free with overwhelming probability. We will first analyze the strength of our scheme's security in terms of a single user's privacy as defined above; then, we will discuss the scenario of a simultaneous attack against multiple users.

THEOREM 4.2. *The proposed scheme in Sec. 3 is $(\tau, 1 - \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \left(1 - \frac{k}{2^q-\tau}\right))$-private for any single user's fingerprint template in the random oracle model.*

Proof. In the random oracle model, the distribution of the hash digests $H(Q_1 \circ \mathit{salt}), \ldots, H(Q_k \circ \mathit{salt})$ is computationally indistinguishable from k random values sampled uniformly from the hash domain, i.e.

$$(r_1, \ldots, r_k) \leftarrow (\{0,1\}^{256})^k$$

Hence, the hash digests do not reveal any dependency of its corresponding input values (triplets of triangles), and the only way that an adversary can get a correct triplet of triangles is through an exhaustive search; that is, by inputting every possible value into the hash function and comparing the output with the target hash digest candidates in the database. Assuming each hash evaluation takes 1 unit time, we have

$$\begin{aligned} P_{win}(\tau) &= \Pr[hit] \cdot 1 + (1 - \Pr[hit]) \cdot \Pr[guess] \\ &= \sum_{i=1}^{k} \frac{\binom{k}{i}\binom{2^q-k}{\tau-i}}{\binom{2^q}{\tau}} + \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \frac{k}{2^q - \tau} \\ &= 1 - \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \left(1 - \frac{k}{2^q - \tau}\right) \end{aligned}$$

where $\Pr[hit]$ is the probability that the adversary can get at least a match within τ attempts (which by the hypergeometric distribution is given by $\sum_{i=1}^{k} \frac{\binom{k}{i}\binom{2^q-k}{\tau-i}}{\binom{2^q}{\tau}}$) and $\Pr[guess]$ is the probability of correctly guessing a member of the set if none of the hashes match (which is $\frac{k}{2^q-\tau}$ since there are k correct choices out of $2^q - \tau$ possibilities). □

Theorem 4.2 gives a tradeoff between τ and ϵ (as in the security definition). However, since there are many parameters involved, next we combine both of these to compute the *expected* time before an adversary can break our system.

COROLLARY 4.3. *The adversary can break the system only in at least* $\frac{2^{q-4}}{k}$ *steps in* expectation.

Proof. Note that the expected number of steps before the adversary "wins" is lower bounded by $P_{win}(\tau) \cdot \tau$ for any $\tau \geq 1$. We show in Appendix A that for $\tau = 2^{q-2}/k$, we have $P_{win}(\tau) \geq 1/4$, which leads to the claimed result. □

Furthermore, a unique 256-bit salt is assigned to each user during enrollment and is stored on the server in plaintext. By appending such salts to the end of the triplets of triangles before hashing, we ensure the orthogonality of the search space among different users when the adversary tries to recover multiple users' fingerprint templates simultaneously. Note that, without such salts, one evaluation of SHA-256 can be used to match all the hash digests of all the users in the entire database. Thus, the adversary might gain tremendous advantage by doing this. By appending the salt, search effort spent on one user's case does not help the adversary attack the other users' fingerprint templates. Therefore, we have reduced the scenario of attacking multiple users to the scenario of attacking a single user.

Another subtle issue is that, in reality, the distribution of the fingerprint templates is not uniform, and each fingerprint minutia point has very limited entropy (bits). As such, the adversary is motivated to brute force fingerprint minutia points and derive triplets of triangles from the guessed fingerprint minutia points instead of directly brute-forcing the triplets of triangles. To prevent such attacks, our scheme filters out all the triplets of triangles that are derived from less than v minutia points, where v is the system-level security parameter, say $v = 7$. The optimal choice of v depends on the other system parameters; we seek to ensure that each triplet of triangles is derived from minutia points that contain at least m bits of entropy. If attacking the minutia points directly leads to a higher P_{win} in the previous analysis, then this attack could prove more effective than the one above. Specifically, if we are not careful in our choice of m and v, the system is

$$(\tau, max(1 - \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \left(1 - \frac{k}{2^q - \tau}\right), 1 - \frac{\binom{2^{mv}-k}{\tau}}{\binom{2^{mv}}{\tau}} \cdot \left(1 - \frac{k}{2^{mv} - \tau}\right)) - \text{private}$$

This analysis is an upper bound that assumes all k hashes are derived from triplets of triangles made from exactly v minutiae points and uses the same analysis as before, but guessing minutiae instead of triplets of triangles. There are typically much fewer than k such triplets making the actually security better than this bound.

Note that the above implies that the security of our system is given by the following version of Corollary 4.3

COROLLARY 4.4. *The adversary can break the system only in at least* $\frac{2^{\min(mv,q)-4}}{k}$ *steps in* expectation.

In our experiments, we will choose parameters so that

$$mv \geq q.$$

4.3 Multiple database scenarios

One advantage of our proposed scheme over the conventional fuzzy value based schemes is the sustainability of our system's security if the authentication scheme is deployed on multiple different servers. In particular, it is well known that the security of the fuzzy vault system may be breached if the same user enrolls with multiple authentication servers. However, as long as the uniqueness of the 256-bit salt is guaranteed, our authentication scheme in the multiple database scenario maintains the same security level as it does in the single database scenarios. Due to the birthday theorem, it is easy to see that the collision probability of the 256-bit salts is negligible for any number of users in any foreseeable real-world application.

5. EXPERIMENTAL RESULTS AND DISCUSSION

We begin with a discussion of experimental results achieved by the hash scheme, which calculates scores by exact matching on sets of SHA-256 hashes of triplets of triangles. We show False Acceptance Rate (FAR) vs False Rejection Rate (FRR) curves for all four databases from the FVC 2002 competition with parameters that were derived experimentally, as well as requiring all triplets of triangles to contain at least 7 distinct minutia points (as is necessary for security as discussed in Section 4). We also present the security at the given parameter set. We then analyze the significance of choosing $4-9$ minimum distinct minutia per triangle triplet and its effects on matching performance through a graph of ROC curves. We show that this choice does have some impact on matching performance for the given set of binning parameters. However this effect is small in comparison to the loss of security due the lack of initial entropy for $v = 4$ (see Table 2 for details).

We also show timing results for the average time for a single match scoring between two fingerprints, both with and without appending the user's salt to the quantized triangle triplets and hashing them with SHA-256. Doing this costs 1.7 milliseconds, a negligible amount of time.

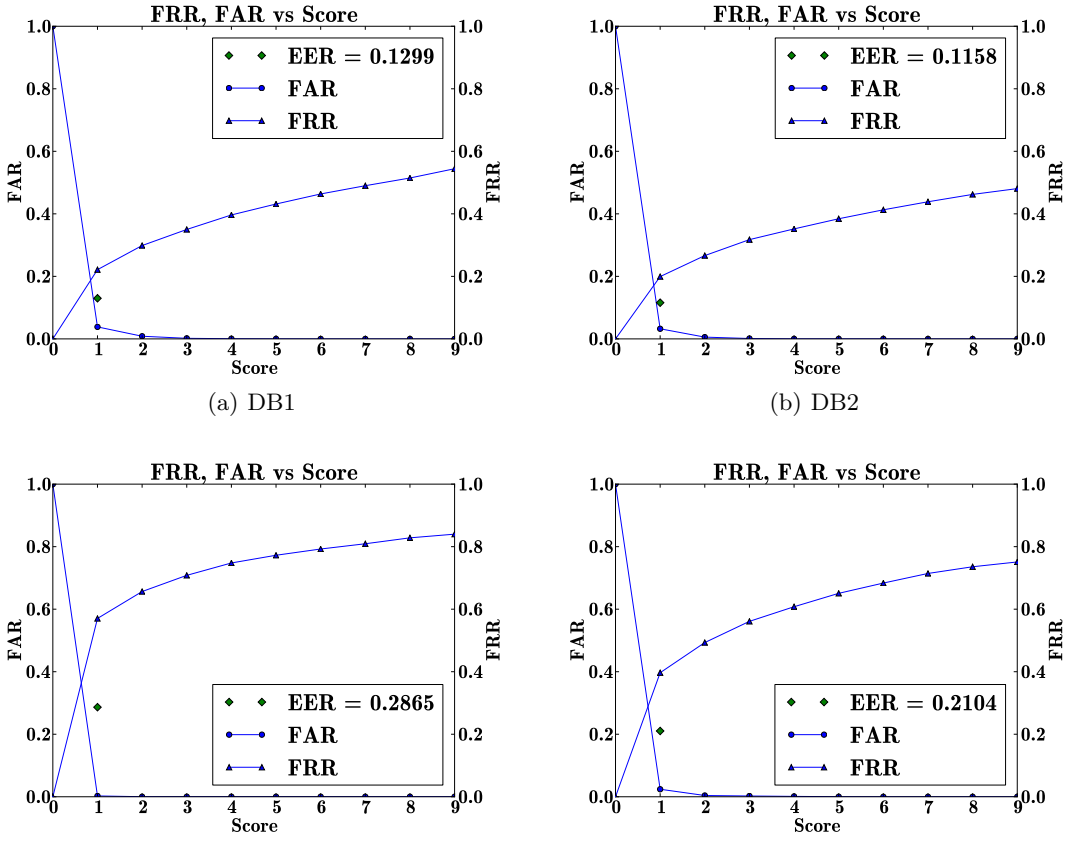
We then present experimental results using an updated fuzzy vault scheme on triplets of triangles from Section 3. We show that for fuzzy vault on triplets of triangles we achieve comparable matching numbers, but at the expense of security and the additional space required to store millions of chaff points. Fuzzy vaults also have security issues when the same fingerprint is stored in multiple databases. This is because an attacker could compare vaults (which use random chaff points) and greatly decrease the number of chaff points by taking the intersection of the two vaults.

We conducted the experiments on the four fingerprint databases from the Second International Fingerprint Verification Competition (FVC2002/DB1-DB4) by finding large curvature points on the contours extracted from the binarized fingerprint images.[19] Additionally, to improve the performance of minutia extraction, the fingerprints were enhanced by the short time Fourier transform algorithm.[20] We also remove suspect spurious minutia points by removing all points within a euclidean distance of 5 from each other. A standard testing protocol for calculating all possible 2800 genuine (100 persons with $\frac{8\cdot 7}{2}$ matches) and 4950 (1 print of each person matched against 1 print of another, or $\frac{100\cdot 99}{2}$) impostor matches has been used.

5.1 Hash Scheme with Triplets of Triangles

Experimental Parameter Results across Databases: Here we present False Acceptance Rate (FAR) versus False Rejection Rate (FRR) plots for all four FVC2002 databases using the same parameters for each. The parameters we used are five x bins, five y bins, and five θ bins as well as a minimum of seven minutia per triangle triplet. In our setting, we have $q = 63$ and $k = 21700$, which by Corollary 4.3 implies that the expected number of steps that an adversary needs to make is at least $\frac{2^{63-4}}{21700} \approx 2^{44.6}$. The binning thresholds for each parameter are variable in width in order to achieve near-uniformity in the quantization. These thresholds were fit to the first database and then used across all four databases.

Traditionally, the Equal Error Rate (EER) is the point where the error rates intersect; however, in this context, this formulation of EERs is not well defined as our scores are discrete and the intersection often falls between these discrete scores. To remedy this, we present the EERs as the average of the FAR and FRR at the first discrete score after the rates cross. Due to the strictness of our system, the FRRs are high but the FARs are low at the EER: see Table 1. This is due to the overall strictness of the system. For these parameters, we find that the threshold score for acceptance would be matching one quantized triangle triplet.



(a) DB1 (b) DB2

(c) DB3 (d) DB4

Figure 3. FAR and FRR graphs for DB1-DB4. We show that the system is relatively strict, with low FARs and high FRRs at the EER, where the threshold score is one. This strictness is caused by the trade off of using more quantization bins for increased security. Less bins can be used, but have been shown to only thin the gap between the two curves at the score 1 (increasing the FAR while decreasing the FRR), and achieve very similar EERs.

Score = 1	DB1	DB2	DB3	DB4
FAR	0.0384	0.0323	0.0026	0.0240
FRR	0.2214	0.1993	0.5704	0.3968

Table 1. Summary of the results of running our algorithm on the FVC2002 databases. In the results presented here, the fingerprints are matched requiring at least 7 points used in the construction of triplets of triangles.

The differences between EERs on the different databases is reflective of the relative difficulty of the databases as seen in Figure 3. Also this choice of parameters is used to increase security, while minding experimental matching performance. Experimentally, we find that more quantization (using fewer

bins) does not improve the EER, but only thins the gap between the FRR and FAR by increasing the FAR and decreasing the FRR. However, less quantization (using more bins) than the above parameters gives us increased security, but at the cost of matching performance. So we settle on these parameters as the optimal tradeoff between performance and security.

Next, in Figure 4, we present a set of ROC curves, using the same set of parameters as above, and show that the choice of minimum number of minutia points per triangle triplet has relatively small effect on matching performance for triplets with 4, 5, or 6 minimum points. Recall from section 4 that the minimum number of points v can affect our security. Since we are using 3 bits each for x, y, and θ, we compute the entropy of a point m to be 9. Since we have $q = 63$, we need $v = 7$ to get the maximum security possible from this field size (recall Corollary 4.4). We do suffer some decrease in security at 4, 5, and 6, though the matching accuracy is not affected much at these values for a fixed set of binning parameters.[†] This allows the implementations to balance the tradeoff between security and matching accuracy as desired.

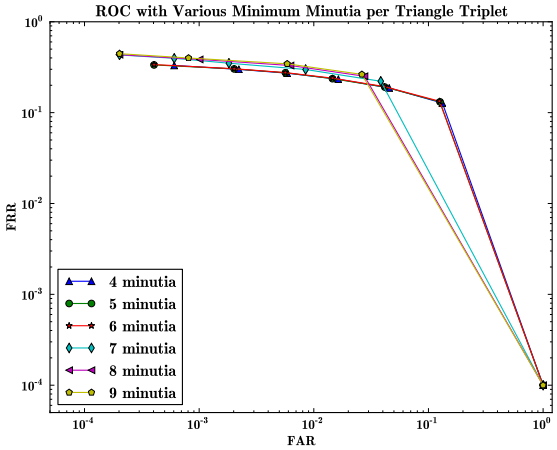


Figure 4. ROC curves for matching performance with the same binning parameters as above, but varying the number of minimum minutia per triangle triplet. We show that the choice of this parameter does not affect performance when 4, 5, or 6 minimum unique minutia points are used, which is desirable. However, as this number increases to 7, 8, or 9, there is a slight negative effect. The sharp drop in the graph is reflective of the very low FRR's, which approach 0 quickly as the score increases. This effect can be seen in figure 3

EER	DB1	DB2	DB3	DB4
$v = 7$	0.1299	0.1158	0.2865	0.2104
$v = 4$	0.1161	0.1029	0.2396	0.1831

Table 2. Summary of the results of running our algorithm on the FVC2002 databases for $v = 7$ and $v = 4$ minimum distinct minutia per triangle triplet. We show that there is some improvement on matching performance when we choose to allow all possible triplets of triangles, but this is not enough to outweigh the loss in security.

Here we show that there is some impact on matching performance when using $v = 4$ instead of $v = 7$ as the minimum number of distinct minutia per triangle triplet. Recall from Section 4 that using a triplet of triangles can be formed by 4, 5, or 6 minutia. Allowing these possible configurations prevents the scheme from achieving maximum security as, in the case of $v = 4$, we have

$$m \cdot v = 9 \cdot 4 = 36 < q = 63$$

[†]Note that we can achieve better matching numbers using fewer minimum distinct minutia, but varying the binning parameters. These results are presented in Appendix B

thus the choice $v < 7$ has a more significant impact on security than matching performance, since the difference in EERs is at most 7%, whereas the difference in initial entropy is about 43% (see Table 2).

Hash Scheme Timing: Here, we show that the extra computation required for appending a user's publicly stored salt and hashing the resulting value using SHA-256 is only 1.7 milliseconds. We average the timing for the scoring function between two fingerprints over a test of the entire database (7750 in total) both with and without appending the salt and hashing the resulting value. For matching by simple exact matching on quantized values $Q \in \mathbb{F}_{2^q}$, we found that the scoring function takes 0.9319 seconds on average. For exact matching on hashes of values computed with appending a user's 256-bit salt to the end of these quantized values, we found that the scoring function takes 0.9336 seconds on average, for a difference of only 1.7 milliseconds per match attempt.

The machine that these timing values are calculated on is an Intel Core i5-3317U CPU, clocked at 1.70GHz $\times$ 4 with 4GB of DDR3 RAM. Currently our program resides exclusively in RAM which does improve data access time. Future work will be conducted to include using an indexed database for data access.

5.2 Fuzzy Vault with Triplets of Triangles

We also ran the fuzzy vault system on triplets of triangles on the FVC 2002 databases. In all cases, this is with the system configured such that z is 1, while the number of chaff points is $100,000$. The matching results are identical to those in Section 5.1. We note that at the score of 1 for the EER, we have no security for fuzzy vault.

However, the security of the system at these parameters is non-existent in practice. Experimentally, we have determined the average number of quantized triplets of triangles over all rotations to be approximately $21,700$ in a single fingerprint, while the value of z that obtains this EER is 1. Recall the formula for fuzzy vault security from (1). This means that, in order to make λ positive, let alone large enough to guarantee reasonable amounts of security, there must exist millions of chaff points (see Section 5.3 for more details). While these chaff points could theoretically be added without significantly impacting the accuracy of the fuzzy vault, doing so over the entire database requires too much memory for such an approach to be practical.

5.3 Discussion and Comparisons

From the results we see that the two schemes from Section 3 are the same with respect to matching accuracy. This is the intuitive result, as in both cases we are in effect finding the set intersection of the enrolled and test fingerprints.

However, where these schemes differ is in their measures of security. For the hash scheme we use standard cryptographic hashes over a large initial field size to achieve security, whereas the fuzzy vault scheme uses chaff points and the degree of the encoding polynomial to achieve security. The main issue with the fuzzy vault scheme is at the optimal matching parameters, where we have a matching threshold of $z = 1$ and tens of thousands of quantized genuine values g. This forces the number of chaff points to skyrocket in order to achieve a positive λ (security) value.

For example, take a fingerprint that, when quantized, becomes represented as a set of $k = 22,000$ quantized triangle triplets (an approximation of the average number, for simplicity). To achieve a $\lambda \geq 0$, this would mean we need

$$0 \leq \sqrt{c \cdot 1} - 22,000,$$

that is

$$c \geq 484,000,000$$

recalling that c is the number of chaff points required in the vault.

The above is *not* the worst-case scenario for the number of elements in the fingerprint set, and also is only for $\lambda \geq 0$. In practice, we would need a much larger λ (say 80). The main issue with ramping up the number

of chaff points per vault is the space required to store this many points is too large. The size of a point in the vault requires 16 bytes. Now adding the the original 22,000 quantized triplets we have

$$484,022,000 * 16 = 7744352000 \text{ B}$$

or

$$\approx 7 \text{ GB per hash!}$$

Now we consider the space requirements of our scheme. A SHA-256 hash of any value produces a digest that is 32 bytes. Using the same approximate number of points as the analysis above, we have

$$22,000 \times 32 = 704,000 \text{ B}$$

or

$$\approx 688 \text{ KB per hash.}$$

Clearly our scheme has the advantage in total storage space needed.

Given the relatively large space requirement of 688 KB for the above hash scheme, we also explored a slightly different version of the hashing algorithm. This one requires only about 64 KB storage space and has the following modifications:

First, we do not rotate the enrolled print and only store the hash set of the triplet of triangles at the original orientation. This gives us our reduction in storage space. For a single rotation of a fingerprint we exerpimentally deteremine the new average set size k to be $k = 1950$. By only storing one rotation, we are storing in total

$$1950 \times 32 \approx 64 \text{ KB}$$

We now modify the algorithm by quantizing the fingerprint at each rotation and taking the hash of only that rotation. We then take the intersection of this with the enrolled print's hash set. We then report our final matching score as the largest intersection encountered over all possible rotations. Therefore, we are effectively matching at only a given rotation instead of across all rotations simultaneously.

Although this method requires only about 10% of the space, we find that the performance and security of the system suffers. We find that the parameters that obtain optimal matching performance for this altered scheme require even further quantization where we quantize $x \in \mathbb{F}_{2^2}$, $y \in \mathbb{F}_{2^2}$, $\theta \in \mathbb{F}_{2^3}$ yielding $q = 51$ and so we are quantizing the resulting triangle triplets into a smaller field $Q \in \mathbb{F}_{2^{51}}$. This means that our level of security decreases (as dictated by Corollary 4.3) to $\frac{2^{q-4}}{k} = \frac{2^{51-4}}{1950} \approx 2^{36.1}$. We also find that matching performance suffers significantly. The performance results for this scheme are available in Appendix C.

With regards to fuzzy vault, let us not forget that there is also the issue of enrolling multiple fuzzy vaults of the same fingerprint in different databases. In this situation, an adversary could simply take the intersection of the two vaults to significantly reduce the number of chaff points (since they are random by design). The proposed hash scheme, which uses SHA-256, does not have these drawbacks and is therefore more secure. With further experimentation and innovation, including the ideas proposed in the future work Section 6, this scheme has the potential to provide real, secure fingerprint matching.

6. FUTURE WORK

Finally, we present some thoughts on how we can improve/build upon the results in this paper.

6.1 Scalability and Indexing

The tests we ran for this work were on databases with only 100 enrollees. However, a real-world system may have millions or billions of users. In this case, our system will scale as it is well-suited for a simple indexing scheme. Since we are storing the hashes of each triplet and performing exact matching, we can store all the hashes for all the users in a single hash table for very efficient lookup during matching. Our later work will explore and test this scalability. Also, our current system resides entirely on local RAM, which limits the size of the database and the total number of users. As such, we have been developing an SQL-based on-disk hash table to allow us to test implementations with large numbers of users.

6.2 Uniform Randomness

Getting the fingerprint template points to be uniform in the field has often hindered our efforts. We currently use non-uniform-width bins during the quantization step so that the values are close to a uniform distribution, but this method is time-consuming and not exact. We are currently developing a method that will apply a separate, random permutation to each enrollee's fingerprint template after quantization. Since the permutation is different for each user, the resulting points in the database will be uniformly distributed in the field, solidifying our security proofs. The primary challenge in implementing this system is to apply the permutation consistently for each matching attempt without leaking any information and without having the user store the permutation as a secret.

6.3 Increased Field Size

We will also investigate more ways of increasing the field size without sacrificing matching accuracy. As described in Section 4, the security of this scheme necessitates a large field size. We achieve this by using triplets of triangles as our fingerprint template points, but we would like to enlarge it further. One idea that we will pursue is to use an arbitrary n-gon instead of exclusively using triangles. If we can increase the size of these polygons, they will have more entropy and we can therefore represent them using more bits. In such an arbitrary system, the level of security could be adjusted by altering n, providing for a tradeoff between security and matching accuracy.

6.4 Security enhancement

Security strength of our proposed scheme depends on many system parameters, so it is correlated with other system properties such as FRR, FAR, EER, etc. (c.f. Sec. 4.) The system should provide an interface that allows us to adjust the security strength arbitrarily. We plan to implement and benchmark the following security enhancement mechanism. Define $H^\lambda := \underbrace{H \circ H \circ \cdots \circ H}_{\lambda \text{ times}}$. We can replace $H(x)$ in our scheme with $H^\lambda(x)$, which makes the adversary's attack λ times slower. Note that this mechanism does not increase the authentication server's complexity (which is essential), because the hashing step is performed at the client side.

This scheme would potentially allow us to move from Corollary 4.3 as our security benchmark and talk about a general (τ, ϵ)-private schemes instead of our current single value that talks about the security of our system in expectation.

6.5 Increased Chaff for Fuzzy Vault

As mentioned above, one issue in using the fuzzy vault when using a system such as this, which has a large amount of data enrolled, is that generating sufficient chaff to guarantee security is difficult due to time and memory issues. One possible solution is to use a pseudo-random number generator to determine the locations and values of the chaff points. This way, one could simulate having sufficiently many chaff points to guarantee security without actually generating all of them. However, this would necessitate revisiting the security argument.

7. ACKNOWLEDGMENTS

This research is supported by NSF grant TC 1115670. Authors Tom Effland, Mariel Schneggenburger, and Jim Schuler are supported by the NSF CSUMS grants 0802994 and 0802964.

REFERENCES

[1] Campisi, P., [*Security and Privacy in Biometrics*], Springer (2013).

[2] Ferrara, M., Maltoni, D., and Cappelli, R., "Noninvertible minutia cylinder-code representation," *Information Forensics and Security, IEEE Transactions on* **7**(6), 1727–1737 (2012).

[3] Scheirer, W. and Boult, T., "Cracking fuzzy vaults and biometric encryption," in [*Biometrics Symposium, 2007*], 1 –6 (sept. 2007).

[4] Nandakumar, K., Jain, A. K., and Pankanti, S., "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Transactions on Information Forensics and Security* **2**(4), 744–757 (2007).

[5] Jain, A. K. and Maltoni, D., [*Handbook of Fingerprint Recognition*], Springer-Verlag New York, Inc. (2003).

[6] Hartloff, J., Bileschi, M., Tulyakov, S., Dobler, J., Rudra, A., and Govindaraju, V., "Security analysis for fingerprint fuzzy vaults," in [*Proc. SPIE*], **8712**, 871204–871204–12 (2013).
[7] Hartloff, J., Dobler, J., Tulyakov, S., Rudra, A., and Govindaraju, V., "Towards fingerprints as strings: Secure indexing for fingerprint matching," in [*Biometrics (ICB), 2013 International Conference on*], 1–6 (2013).
[8] Germain, R., Califano, A., and Colville, S., "Fingerprint matching using transformation parameter clustering," *IEEE Computational Sci. and Engineering* **4**(4), 42–49 (1997).
[9] Choi, K., Lee, D., Lee, S., and Kim, J., "An improved fingerprint indexing algorithm based on the triplet approach," in [*Audio and Video based Biometric Person Authentication (AVBPA)*], (2003).
[10] Bhanu, B. and Tan, X., "Fingerprint indexing based on novel features of minutiae triplets," *IEEE Pattern Analysis and Machine Intelligence* (May 2003).
[11] Juels, A. and Sudan, M., "A fuzzy vault scheme," *Des. Codes Cryptography* **38**(2), 237–257 (2006).
[12] Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A., "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.* **38**(1), 97–139 (2008).
[13] Jea, T.-Y. and Govindaraju, V., "A minutia-based partial fingerprint recognition system," *Pattern Recognition* **38**(10), 1672–1684 (2005).
[14] Bhanu, B. and Tan, X., "Fingerprint indexing based on novel features of minutiae triplets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **25**(5), 616–622 (2003).
[15] Chang, E.-C., Shen, R., and Teo, F. W., "Finding the original point set hidden among chaff," in [*Proceedings of the 2006 ACM Symposium on Information, computer and communications security*], *ASIACCS '06*, 182–188, ACM, New York, NY, USA (2006).
[16] Nagar, A., Nandakumar, K., and Jain, A. K., "Securing fingerprint template: Fuzzy vault with minutiae descriptors," in [*ICPR*], 1–4 (2008).
[17] Kumar, G., Tulyakov, S., and Govindaraju, V., "Combination of symmetric hash functions for secure fingerprint matching," in [*20th International Conference on Pattern Recognition (ICPR)*], 890 –893 (aug. 2010).
[18] Scheirer, W. J. and Boult, T. E., "Cracking fuzzy vaults and biometric encryption," in [*Proceedings of Biometrics Symposium*], 1–6 (2007).
[19] Govindaraju, V., Shi, Z., and Schneider, J., "Feature extraction using a chaincoded contour representation of fingerprint images," in [*4th international conference on Audio- and video-based biometric person authentication*], **2688**, 268–275, Springer-Verlag, Guildford, UK (2003).
[20] Chikkerur, S., Cartwright, A. N., and Govindaraju, V., "Fingerprint enhancement using STFT analysis," *Pattern Recognition* **40**(1), 198–211 (2007).

APPENDIX A. A CALCULATION

LEMMA A.1. *For any* $1 \le k \le 2^{q-2}$ *and* $\tau \le \frac{2^{q-2}}{k}$*, we have*

$$1 - \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \left(1 - \frac{k}{2^q - \tau}\right) \le \frac{3}{4}. \tag{2}$$

Proof. Note that (2) is satisfied if

$$\frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \cdot \left(1 - \frac{k}{2^q - \tau}\right) \ge \frac{1}{4}.$$

Since $\tau \le 2^{q-1}/k \le 2^{q-1}$, we have $\left(1 - \frac{k}{2^q-\tau}\right) \ge 1 - \frac{k}{2^{q-1}} \ge \frac{1}{2}$, where the last inequality follows from the assumption that $k \le 2^{q-2}$. This implies that the above bound is satisfied if

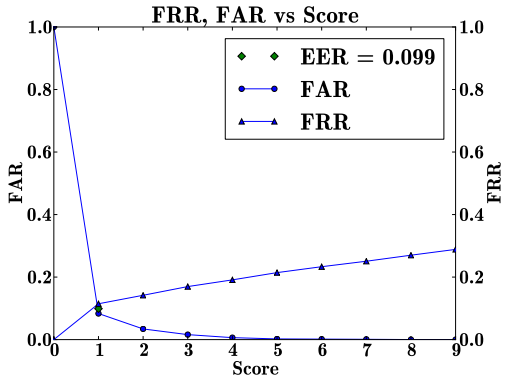
$$\mathcal{B} \stackrel{\text{def}}{=} \frac{\binom{2^q-k}{\tau}}{\binom{2^q}{\tau}} \ge \frac{1}{2}.$$

Now note that

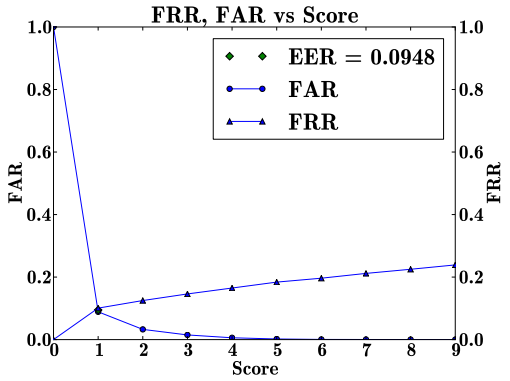
$$\begin{aligned}
\mathcal{B} &= \frac{(2^q-k)!(2^q-\tau)!}{(2^q-k-\tau)!(2^q)!} \\
&= \frac{(2^q-\tau)(2^q-\tau-1)\cdots(2^q-\tau-(k-1))}{2^q(2^q-1)\cdots(2^q-(k-1))} \\
&= \prod_{i=0}^{k-1}\left(1-\frac{\tau-i}{2^q-i}\right) \\
&\geq \left(1-\frac{\tau}{2^q-k+1}\right)^k \\
&\geq \left(1-\frac{\tau}{2^{q-1}}\right)^k \\
&\geq 1-\frac{\tau k}{2^{q-1}} \\
&\geq \frac{1}{2},
\end{aligned}$$

as desired. In the above the first inequality follows by noting that for $0 \leq i \leq k-1$, $\tau - i \leq \tau$ and $2^q - i \geq 2^q - k + 1$. The second inequality follows by our bound on k. The third inequality follows since all the parameters are positive integers and the final inequality follows since $\tau \leq 2^{q-2}/k$. □

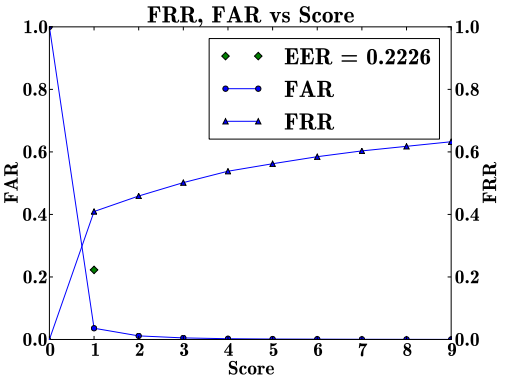
APPENDIX B. EXPERIMENTAL RESULTS USING FIVE DISTINCT MINUTIA POINTS PER TRIPLET WITH ALTERNATE BINNING



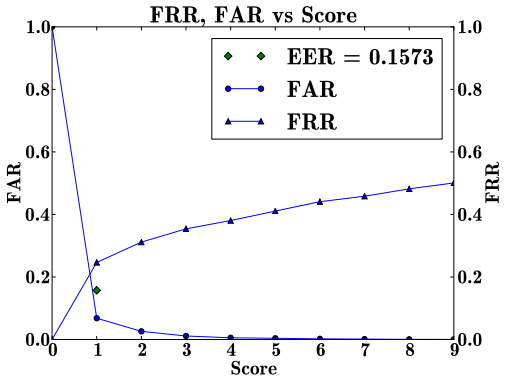
(a) DB1



(b) DB2



(c) DB3



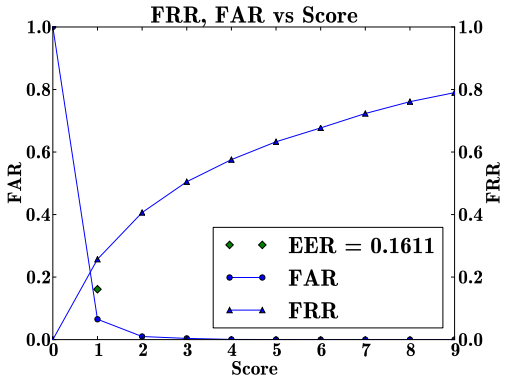
(d) DB4

Score = 1	DB1	DB2	DB3	DB4
FAR	0.0834	0.0889	0.0360	0.0681
FRR	0.1146	0.1007	0.4093	0.2464

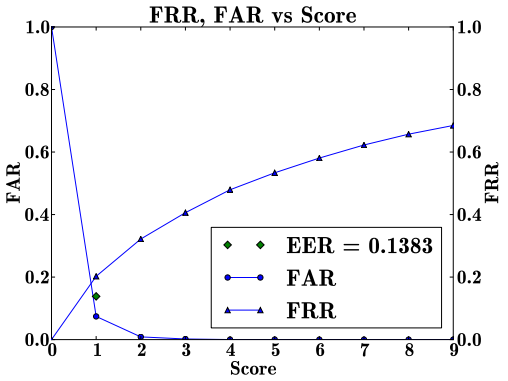
Table 3. Summary of the results using five minimum distinct minutia points with alternate binning on the FVC2002 databases. This shows that the matching numbers do improve; however, this is with a loss of security.

Here we present results that produce better matching performance with an alternate binning scheme. In these experiments, the number of minimum distinct minutia per triplet of triangles v was relaxed from 7 to 5. This allows us to decrease the quantization and use more bins. Here we use six x bins, six y bins, and eight θ bins. Although the matching performance here is better and we have the same field size $q = 63$, the initial entropy $m \cdot v = 9 \cdot 5 = 45 < q = 63$ and is not high enough to fullfill the security potential of the scheme. From this we can glean that performance can be improved, but at the cost of security of the system.

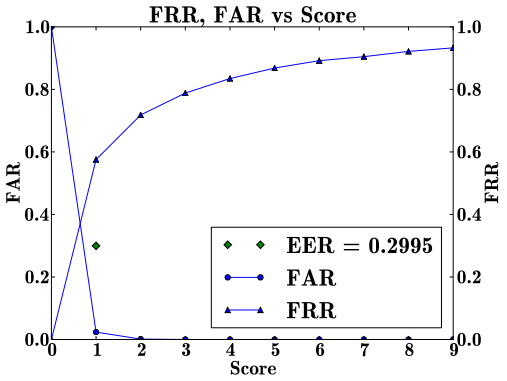
APPENDIX C. EXPERIMENTAL RESULTS OF ALTERNATE SPACE-SAVING HASH SCHEME



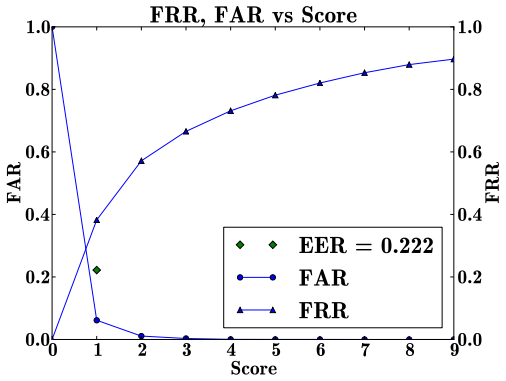
(e) DB1



(f) DB2



(g) DB3



(h) DB4

The FAR and FRR plots for the altered space-saving scheme appear in Figure C while Table C refer to specific FAR and FRR values at the EER score of 1.

The matching performance as shown through these figures, as well as the weakended security strength shown at the end of Section 5 are enough to see that this system, while requiring signifantly less space, leaves something to be desired.

Score = 1	DB1	DB2	DB3	DB4
FAR	0.0653	0.1180	0.0236	0.1042
FRR	0.2571	0.1829	0.5753	0.3404

Table 4. Summary of the results of running our altered, space-saving algorithm on the FVC2002 databases. In the results presented here, the fingerprints are matched requiring at least 5 points used in the construction of triplets of triangles.