Turk J Elec Eng & Comp Sci, Vol.18, No.4, 2010, © TÜBİTAK
doi:10.3906/elk-0910-228

Securing fuzzy vault schemes through biometric hashing*

Cengiz ÖRENCİK, Thomas B. PEDERSEN, Erkay SAVAŞ, Mehmet KESKİNÖZ
Faculty of Engineering & Natural Sciences, Sabancı University, İstanbul, 34956, TURKEY
e-mail {cengizo@, pedersen@, erkays@, keskinoz@}sabanciuniv.edu

Abstract

The fuzzy vault scheme is a well-known technique to mitigate privacy, security, and usability related problems in biometric identification applications. The basic idea is to hide biometric data along with secret information amongst randomly selected chaff *points during the enrollment process. Only the owner of the biometric data who presents correct biometrics can recover the secret and identify himself. Recent research, however, has shown that the scheme is vulnerable to certain types of attacks. The recently proposed "correlation attack", that allows linking two vaults of the same biometric, pose serious privacy risks that have not been sufficiently addressed. The primary aim of this work is to remedy those problems by proposing a framework based on distance preserving hash functions to render the correlation attack inapplicable. We first give definitions which capture the requirements such hash functions must posses. We then propose a specific family of hash functions that fulfills these requirements and lends itself to efficient implementation. We also provide formal proofs that the proposed family of hash functions indeed protects the fuzzy vault against correlation attacks. We implement a* hashed *fuzzy vault using fingerprint data and investigate the effects of the proposed method on the false accept and false reject rates (FAR and FRR, respectively) extensively. Implementation results suggest that the proposed method provides a complete protection against correlation attacks at the expense of small degradation in the FRR.*

Key Words: *Fuzzy vault, Biometrics, Biometric hashing, Fingerprint, Privacy*

1. Introduction

Identification for access control and other purposes can be achieved by utilizing three factors: 1) what you know (e.g. passwords), 2) what you have (e.g. smartcards), and 3) what you are (biometric data identifying a person). Either these factors can be used alone or any combination of the three can be used together to increase security and compensate for the weaknesses of one factor. While passwords can be forgotten and smartcards can be stolen, a biometric is inseparable from an individual and always accessible providing a comparably high

*Part of this work is presented in SECRYPT 2008 [1] level of security. In addition, it can easily be combined with other factors to increase security further. Biometric identification, on the other hand, also suffers from two major drawbacks: 1) the noisy nature of the biometric measurement process and 2) privacy issues due to the fact that biometric data reveals private information about the individuals which is not intended to be revealed otherwise.

Juels and Sudan [2] proposed the so-called *fuzzy vault* scheme to overcome these two drawbacks associated with the usage of biometrics for identification. The main idea in the fuzzy vault scheme is to combine error correction and secret sharing where the biometric data, together with a *secret*, defines a codeword of an error correction code. Given the fingerprint, the codeword can be corrected, and the secret extracted. However, the secret itself does not reveal anything about the biometric data. If the secret is compromised, one can always choose another secret to combine with the same biometric.

To extract the secret information for the verification process there are two popular approaches. One approach, proposed by Clancy et al. [3], is to use a variant of the Reed-Solomon (RS) decoding algorithm [2]. The other approach is to use brute-force, where different subsets of measured minutiae points are used to reveal the secret polynomial. Uludag et al. [4] successfully apply the brute-force method to the fuzzy vault scheme for fingerprint biometrics. Although both the brute force decoding and the RS decoding methods reveal the same secret polynomial, their time complexities are different, which is explored in [1].

In this paper we first address implementation details of the two aforementioned methods for the fuzzy vault scheme, namely the brute-force and RS decoding methods, and lay out a guideline for implementing the fuzzy vault scheme efficiently for biometric identification.

It is well-known that the fuzzy vault scheme is vulnerable to so-called "correlation attacks" [5] [6] where two distinct fuzzy vaults for the same fingerprint can easily be linked. This attack, therefore, poses a serious security and privacy risk. As a major contribution of this work, we next develop a framework to achieve a secure implementation of fuzzy vaults under correlation attacks. Specifically, in this framework, we propose a biometric hash function, and prove that it provides security against the correlation attack.

The contributions of this work can be summarized as follows:

- After we explain the correlation attack in Section 3.1, we propose a framework based on distance preserving hash function families to protect the fuzzy vault against the attack. The basic idea is to store the hashed points, instead of the original vault points themselves.

- In the framework we outline the requirements that a suitable family of hash functions fulfill for a provable protection against the correlation attack. We give three definitions for the properties of a candidate family of hash functions; namely, robustness, non-linkability, and non-invertability.

- We propose a specific family of hash functions, which preserves distance and fulfills the aforementioned requirements. The proposed family of hash functions are efficiently computable and suitable for biometric applications. We also present the security analysis of the family of hash functions and prove that the correlation attack is inapplicable to the fuzzy vault scheme if the proposed family of hash functions is used.

- We furthermore study the effects of vault size and hash usage on the security and performance of the fuzzy vault in terms of false accept and false reject rates as well as on timing of verification stage.

- As a minor contribution, we provide a complete guideline for efficient and secure implementation of fuzzy vault for biometrics.

The most important characteristics of the proposed hash based framework is the fact that it requires no other secret values to be stored to protect the fuzzy vault. A previous attempt by Nandakumar et al. [7] is basically a two-factor authentication system where users are required to use *secret* passwords in addition to their biometric. While providing a sufficient security against correlation attack, it suffers from the well-known security issues and inconveniences associated with using passwords:

1. User has to remember a secret password which is frequently forgotten and compromised.
2. Passwords do not provide sufficient security and easily succumb to cryptanalysis. If the password is compromised, the system becomes vulnerable to correlation attacks.

However, in the proposed scheme neither the user nor the database owner has to remember any secret value. Hash parameters are chosen randomly and made public as they need to be in a hash-based system. Our security analysis proves that no feasible attack can compromise the biometric.

The remainder of this paper is organized as follows. For the sake of completeness, Section 2 reviews the fuzzy vault scheme and explains the details of the brute force and Reed Solomon decoding algorithms [8] used for reconstructing the secret polynomial hidden in the fuzzy vault. This section also provides a summary of comparative analysis for the efficiencies of two techniques used for polynomial reconstruction. In Section 3, the three previously proposed attacks targeted on the *fuzzy vault* scheme, namely the location based attack [1], the brute force attack [3], [9] and the correlation attack [5], are revisited. In Section 4, we propose keeping hash values of the minutia points instead of the minutia points themselves for securing the scheme against correlation attack. We define the requirements that a hash function satisfy to be used in a secure *fuzzy vault* scheme. We also propose a special family of hash functions and present proofs that our proposed hash function family complies with our definitions (see Appendix for the proofs). Section 5 explores the effects of the vault sizes and use of the proposed hash function on the performance and security of the fuzzy vault using experimental data. Section 6 is devoted to our conclusions and the summary of the work.

2. Review of fuzzy vaults from fingerprints

In this section we give a brief outline of the techniques used in the application of the fuzzy vault scheme to fingerprint biometrics. In the enrollment stage, a fuzzy vault is created by combining a secret with the fingerprint of the user to form a set of points. The fuzzy vault hides the points created from the fingerprint and the secret by adding random points to the vault. The points are created in a way that the secret can be obtained if the right fingerprint is provided during verification. The verification stage consists of two phases: 1) alignment of the measured fingerprint to the points in the fuzzy vault, and 2) the reconstruction of the secret.

The enrollment and alignment stages and the two most frequently used decoding methods are described briefly in the following sections. Note that these stages outline our implementation and may differ from other *fuzzy vault* implementations.

2.1. Enrollment stage

During the enrollment stage n minutiae points from the fingerprint of the user is presented to the system. Each minutia point is a point $(x_i, y_i) \in \mathbb{Z}_{2^w}^2$ with two w-bit coordinates, which we concatenate to form the $2w$-bit integers $\overline{x}_i = x_i || y_i \in \mathbb{Z}_{2^{2w}}$.

Now, a $2kw$-bit secret key S, used for identification, is divided into k equal length parts, $\{S_1, \ldots, S_k\}$. From these parts we define the secret polynomial $p(x) = \sum_{i=1}^{k} S_i x^i$ in $\mathbb{Z}_q[x]$ of degree $k-1$ where q is a prime larger than any $2w$-bit number. Since the secret polynomial has degree $k-1$, k points that lie on this polynomial are sufficient to successfully reconstruct the polynomial.

For each of the $2w$-bit number $\overline{x}_i$ formed from the concatenation of coordinates of a minutiae point, the secret polynomial $p(x)$ is evaluated in $\mathbb{Z}_q$ to define $\overline{y}_i = p(\overline{x}_i)$, for $i \in \{1, \ldots, n\}$. The n resulting points $(\overline{x}_i, \overline{y}_i) \in \mathbb{Z}_q^2$ are denoted the *genuine points* of the fuzzy vault.

Clearly both the minutiae points of the original fingerprint and the secret polynomial can be recovered from the genuine points in the fuzzy vault. To protect the secret key and the genuine points we pick *chaff points* at random from $\mathbb{Z}_q^2$ and add them to the fuzzy vault. The chaff points have to be chosen such that their x coordinates are at distance at least t from $\overline{x}_i$-coordinates of any of n genuine points. Chaff points are added in this fashion until the fuzzy vault contains a total of C points (for some $C \gg n$). The resulting fuzzy vault with C points is assumed to be accessible to anyone including an external attacker. Since the attacker cannot tell the difference between genuine points and chaff points, he cannot easily find either the original minutia points or the secret key. The block diagram of the original *fuzzy vault* enrolling scheme is shown in Figure 1.

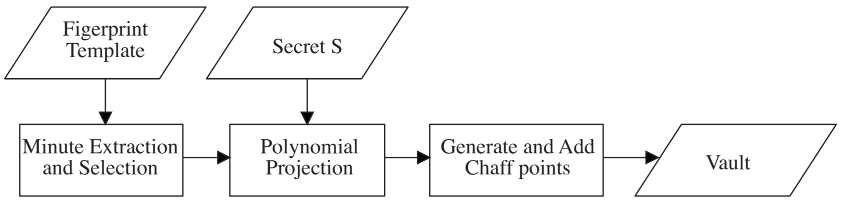
Figerprint Template
Secret S
Minute Extraction and Selection
Polynomial Projection
Generate and Add Chaff points
Vault

Figure 1. Block diagram for enrollment.

Since the polynomial reconstruction methods may return an incorrect polynomial, in the enrollment phase the implementation in [4] adds a cyclic redundancy check (CRC) of the secret S as a coefficient of the secret polynomial to verify that the correct polynomial is found in the verification stage. However, in our implementation we instead check if there are at least $k + \mu$ vault points lie on the polynomial, for some $\mu > 1$, to verify that the correct polynomial is found. Adding redundant information such as the CRC in [4] is likely to be utilized in exhaustive search attacks to check whether the correct polynomial is constructed. However, since our method utilizes the *already available* redundancy (i.e. more points lie on the polynomial than its degree) to construct the secret polynomial, it does not necessarily deteriorate the strength of the systems. In our tests we see that both methods give the same False Accept Rate (FAR) and False Reject Rate (FRR) results (Section 5).

2.2. Verification stage

Given the original minutiae points, the genuine points in the fuzzy vault can be found by matching the $\overline{x}_i$ values in the vault with those of the minutia points. The owner of the fingerprint can always provide the original minutiae points up to a rotation and translation, and thus find some of the genuine points in the fuzzy vault. However, due to noise in the measurement process the user may not find all genuine points, and may even mistake some of the chaff points for genuine points. It is the relationship between polynomial reconstruction and error correction that allows the authentic user to recover the secret polynomial, and thus the secret key, as long as he presents a majority of the genuine points. For an attacker, on the other hand, it is difficult to reconstruct the secret polynomial without knowing the original biometric data, since the fuzzy vault contains much more chaff points than genuine points.

The goal of the verification stage is to reconstruct the secret polynomial from the genuine biometric data, and thus enable us to recover the secret key S. The recovered secret key is then used for identification. When a user presents a genuine fingerprint for identification, $m \leq n$ minutiae points are expected to match the points in the fuzzy vault. The person does not have to present the same set of minutiae points for each verification process. This is a necessary feature of biometric authentication since the fingerprint measurement is a noisy process: in each verification a different set of measured minutiae points will match the points in the *fuzzy vault*. The block diagram of the original *fuzzy vault* verification scheme is shown in Fig 2. In the following sections, we explain the steps of the verification stage in detail.

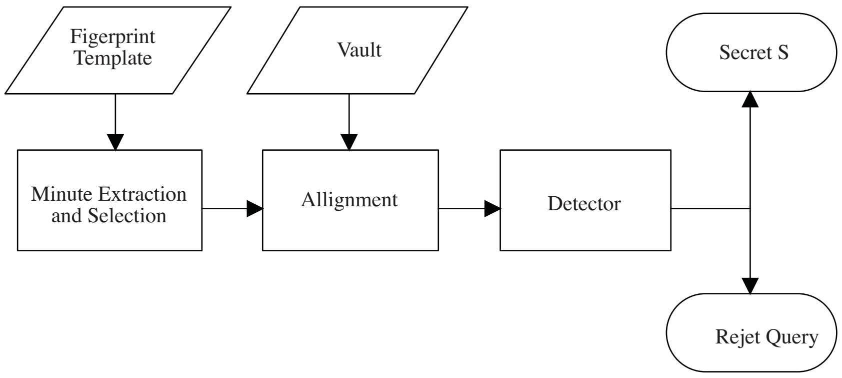

Figure 2. Vault verification.

2.3. Alignment phase

Since a user who wishes to authenticate himself with his fingerprint may inadvertently rotate and translate his fingerprint at each measurement, the verification process needs to pre-process the fingerprint before applying the polynomial reconstruction algorithm. The pre-processing tries to align the query fingerprint to the enrolled fingerprint stored in the fuzzy vault. At the end of the alignment phase, the $\overline{x}_i$ coordinates in the fuzzy vault are matched with minutiae points from the query fingerprint, and the corresponding fuzzy vault points are presented to the system for verification. This set will consist of some of the genuine points, but also some chaff points. This is known as the *verification list* and when it contains at least $k + \mu$ genuine points, the secret polynomial can be reconstructed.

Aligning two fingerprints is a difficult task and errors in this phase could lead to false rejects. There are several different approaches for fingerprint alignment [4], but we use alignment by exhaustive search. In this method, several combinations of translations in x and y coordinates plus rotations of the points of the query fingerprint are compared with the points in the vault. If a point in the query fingerprint is closer to a fuzzy vault point than a certain number of pixels[1] we say that the two points are *matched*. This process is repeated for many different combinations of translations and rotations and the combination with the maximum number of matching points is the output of the alignment phase.

[1]The matching is done in the (x, y)-plane of the fingerprints, not in the $(\overline{x}, \overline{y})$-plane of the fuzzy vault.

2.4. Brute force decoding approach

To reconstruct the secret polynomial we need k genuine points. In the brute force approach we try all combinations of k out of the m matching points. Note that some of the m matching minutiae points will be chaff points from the *fuzzy vault.* However, if k genuine points are used during the exhaustive search, the scheme will be successful.

In the brute force approach, k pairs of $(\overline{x}_i, \overline{y}_i)$ are chosen randomly from the verification list and the polynomial on which the selected k pairs lie is calculated using Lagrange interpolation [10]. If μ other fuzzy vault points lie on the same polynomial, the fingerprint is accepted; otherwise it is rejected. If the fingerprint is rejected, another random set of k pairs are taken and the process is repeated. The maximum number of trials is set to a high value, after which the program rejects the fingerprint if no polynomial satisfying the condition is found. The drawback of the brute force approach is high computation complexity when the query fingerprint is too noisy, causing the verification set to contain less genuine points and more chaff points.

2.5. Reed solomon decoding approach

Using error correcting codes for the implementation of the fuzzy vault is first proposed in [2]. The authors state that after the matching minutiae points are obtained, it is more efficient to use a Reed-Solomon (RS) decoder than brute force to reconstruct the secret polynomial. The Reed-Solomon decoders have an error correction capability of $\tau = \frac{m-k}{2}$ errors. Even though the Guruswami-Sudan list decoding algorithm [11, 12] can correct errors beyond this limit ($\tau_{GS} = \sqrt{mk}$), it is not suitable in our case due to efficiency reasons. The best choice is to use the Berlekamp-Massey (BM) algorithm as explained in [3] since it is fast, easy to implement and widely studied.

However, details on how to apply the RS decoding method to fuzzy vaults, are given neither in [2] nor in [3]. These papers mention the decoder, which they name the UNLOCK function, as a black box that reconstructs the secret polynomial given the m matching points. The lack of detailed description of the method caused some misunderstanding in the literature; for instance the authors of [13] claim that the Reed-Solomon decoding is not applicable in the case of the *fuzzy vault* scheme. In the following we give a description of how to use the Reed Solomon codes and Reed Solomon decoding methods in the fuzzy vault scheme.

2.5.1. RS decoding with BM algorithm

As explained in the enrollment stage, when we construct the *fuzzy vault* we evaluate the secret polynomial for all n minutiae points, i.e. $\overline{y}_i = p(\overline{x}_i)$ for $i = 1, \ldots n$. This can be put into a matrix-vector formulation as follows:

$$\left[\begin{array}{cccc} \overline{y}_1 & \overline{y}_2 & \ldots & \overline{y}_n \end{array}\right] = \left[\begin{array}{cccc} p_0 & p_1 & \ldots & p_{k-1} \end{array}\right] \mathbf{G},$$

where the matrix $\mathbf{G}$ is given as

$$\mathbf{G} = \left(\begin{array}{cccc} 1 & 1 & \ldots & 1 \\ \overline{x}_1 & \overline{x}_2 & \ldots & \overline{x}_n \\ \overline{x}_1^2 & \overline{x}_2^2 & \ldots & \overline{x}_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \overline{x}_1^{k-1} & \overline{x}_2^{k-1} & \ldots & \overline{x}_n^{k-1} \end{array} \right).$$

This is indeed a *shortened* Reed Solomon (RS) encoding with the generator matrix $\mathbf{G}$ [8]. It is crucial to notice that the generator matrix changes for each user, which differ from the conventional application of the RS encoding method. Since the enrollment stage essentially utilizes the RS encoding, reconstruction of the secret polynomial in the verification stage can be achieved by employing an *RSDecoder*.

The codeword to decode in the *fuzzy vault* scheme is the evaluation of a polynomial of degree $k-1$ over a set of m distinct points in field F. The codeword consists of m pairs $(\overline{x}_i, \overline{y}_i)$ where $\overline{x}_i \in F$ is the minutiae point that matches either a genuine or chaff point in the *fuzzy vault*. If the codeword satisfies $\overline{y}_i \neq p(\overline{x}_i)$ for less than τ of the given values of i, the decoder returns the secret polynomial $P(x)$ and thus the fingerprint will be verified. Otherwise, the decoder returns a false polynomial.

The RS decoding with Berlekamp Massey (BM) algorithm takes two vectors ($\mathbf{X} = [\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m]$ and $\mathbf{Y} = [\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_m]$) where $\mathbf{X}$ is the code used to create the parity check matrix and $\mathbf{Y}$ is the codeword with errors. The method has four major steps :

1. Computation of canonical parity-check matrix.

2. Computation of syndromes.

3. Computation of error locator polynomial and error locations.

4. Computation of secret polynomial.

The details of these steps are explained in [8].

After these steps, the secret polynomial is reconstructed by using the Lagrange interpolation method with the correct minutiae points if the number of errors does not exceed a security parameter τ. Otherwise, the function returns a wrong polynomial of degree $k-1$. Again we check if more points that lie on the same polynomial exist. If not, the function is called with fewer number of pairs. This process is repeated a couple of times with some of the different random pairs being removed from the list. If the algorithm still returns a wrong polynomial as output after these attempts, then the fingerprint is rejected.

2.6. Computational complexity of polynomial reconstruction

In [1], the computational complexities of the two decoding methods was compared, in order to clarify as to which method is optimal depending on the parameters of m and k where m is the number of matched points and $k-1$ is the degree of the secret polynomial. The complexity of two methods are given in terms of multiplication and multiplicative inverse operations in F_q.

While one trial of Reed Solomon decoding has $4m^2 + m(k^2/3) - m(3k+2) + k^2$ multiplication and m inverse operations, Brute Force decoding has $l(k^2+5m+k)$ multiplication operations, where l is the number of trials needed on the average, which naturally increases with the error in the query fingerprint. Without knowing the number of trials l in the brute force method it is not easy to compare two methods. Comparison is only possible with experiments on real and synthetic data, which was presented in our previous work with trade off curves both for timing and number of required operations [1].

3. Attacks on fuzzy vault

Although Juels and Sudan [2] proved that the *fuzzy vault* scheme satisfies some security properties, it is still vulnerable to some attacks. The attacks on the *fuzzy vault* scheme, mostly assume the interception of a vault from a database. There are several attacks targeted on the *fuzzy vault* scheme such as location based attack [1], brute force attack [3], [9] and the correlation attack [5] which can be applied in reasonable amount of time. Therefore the *fuzzy vault* scheme is insecure without additional security measures.

In our previous work [1], two modifications to the enrollment stage are proposed in order to strengthen the fuzzy vault against location based attack and brute force attack. In [1], it is shown that the locations of fake (chaff) points leak some valuable information and a new chaff point placement technique that prevents that information leakage is proposed. A novel method for chaff point creation that decreases the success rate of the brute force attack from 100% to less than 3.3% is also proposed in [1]. That modification can also be applied to this proposed method with hashing in order to make the vault secure against brute force attack. However, any method that improves the security of fuzzy vault scheme against the correlation attack, which requires no secret value other than the biometric information was not proposed in any of the previous works. In this work, as a remedy against correlation attack, we propose to keep distorted versions of the biometric that preserves the invariants of the biometric image. The details of this method is explained in Section 4.2. For the completeness of the work, the correlation attack is given below.

3.1. Correlation attack

Scheirer et al. [5] proposed a class of attack called *attack via record multiplicity.* The attack assumes that the attacker intercepts at least two *fuzzy vaults* that belong to the same user. Note that these vaults may be created by different secret polynomials and different chaff points but the genuine points should highly overlap since they are the minutia points of the same fingerprint. Scheirer claimed that correlating these two vaults may reveal the biometric data (i.e. minutia points). Later, Kholmatov et al. [6] showed that by using this property of the *fuzzy vault* scheme, it is possible to link an unknown vault to another vault that is constructed by the same biometric in a reasonable amount of time with high probability. Kholmatov et al. [6] calculated that given two matching vaults, the secret polynomial can be revealed 59% of the time. They also showed that for 41% of the cases, they can link an unknown vault to a set of vaults that contains the matching vault. This attack can be done in less than 8 minutes according to their tests.

Due to this reason, some additional security measures are necessary for a secure *fuzzy vault* scheme. Recently Nandakumar et al. [7] implemented the *fuzzy vault* scheme by combining it with passwords. This scheme successfully overcomes the vulnerability of the scheme against correlation attack with a slight increase in the false reject rates. However, this system is a basically two factor authentication scheme where the user has to provide both the password and the biometric during authentication and that may be inconvenient in certain applications.

4. Preventing correlation attacks

In this section, we explain our framework that is based on distance preserving hash function families that can be used to prevent the correlation attack. In our framework all vault points are hashed and the matching is performed on the hashed vault points. The utmost goal of our framework is to provide *correlation resilience*:

Definition 1 (Correlation Resilience) *Let H be a family of hash-functions. Let an attacker have access to a fuzzy vault v, and a database of N vaults where exactly one of them is created from the same biometric information as v, and where all fuzzy vaults are created with the family of hash-functions H. The vault is* correlation resilient *if the probability of the attacker can find the matching vault is*

$$P[right] \leq \frac{1}{N-1} + \psi,$$

where ψ is a security parameter.

We first define the requirements that a hash function should satisfy to be used in a secure *fuzzy vault* scheme. We then propose a specific family of hash function which satisfies these requirements, and which overcomes the vulnerability against correlation attack.

4.1. Requirements of the hash function

Due to the correlation attack against *fuzzy vault*, minutia points should be randomized. One well-known method for randomization is encryption but this requires safeguarding of the private keys. When using encryption anyone who can access the key (including a malicious server) can obtain the original fingerprint, and thus do the matching.

The use of hash values of the minutia points, instead of the minutia points themselves is an efficient randomization method since minutia points cannot be recovered from their hash values. Furthermore, this method does not rely on the secrecy of a private key.

Any hash function that is used for biometrics should satisfy the following properties:

1. (Robustness) Verifying a legitimate user should be possible.

2. (Non-Linkability) It should be secure against correlation attack (i.e. it should not be possible to correlate two vaults created from the same biometric data).

3. (Non-Invertability) It should not be possible to learn the original biometric from the hashed version of it.

In the rest of this paper the following notations will be used. We define a family of hash-functions $H = \{h : \mathbb{Z}_q \to \mathbb{Z}_{p^2}\}$ where $q > p^2$ and represent the hash of a minutia point with $h(x, y)$. The hashed vault space is denoted as $\mathbb{Z}_r^{2C}$ where r is the smallest prime greater than p^2 and there are C points in a vault with area of $r \times r$. The notation (x, y) and (x', y') is used to represent the original minutia coordinates of a vault point (i.e. before taking the hash value). $M_v(w)$ is a probabilistic function that aligns fuzzy vaults w and v, and returns the number of points matched. Matching is performed by using the algorithm explained in 2.3.

Definition 2 (Robustness) *For all hash functions $h \in H$, and all points $(x, y), (x', y') \in \mathbb{Z}_q$.*

$$\|(x, y) - (x', y')\| < \delta \Rightarrow \|h(x, y) - h(x', y')\| < \delta \tag{1}$$

Intuitively, given two very close fingerprint images (i.e. one is the noisy version of the other one), the hash function is robust if their hash values are also very close. In this work, we use Chebyshev distance [14] (also called infinity norm distance) to measure the distance between two points in a vault which can be expressed as the greatest of their differences along x and y coordinates. The Chebyshev distance between two vectors $\mathbf{a}$ and $\mathbf{b}$, with standard coordinates a_i and b_i, respectively, is:

$$\|\mathbf{a}-\mathbf{b}\|_\infty = \max_i(|a_i - b_i|). \tag{2}$$

To define the distance between two vectors in a finite field $\mathbb{Z}_{p^2}$ we use a modified Chebyshev distance which involves modulo operations.

$$\begin{aligned} \|\mathbf{a}-\mathbf{b}\|_\infty &= \max_i(|a_i - b_i|_p), \text{ where} \\ |a_i - b_i|_p &= \min(|a_i \bmod p - b_i \bmod p|, \\ &\quad p - |a_i \bmod p - b_i \bmod p|) \end{aligned}$$

Note that the modified Chebyshev distance satisfies all the distance axioms with the exception that, $\|\mathbf{a}-\mathbf{b}\|_\infty = 0$ if and only if $\mathbf{a} \bmod p = \mathbf{b} \bmod p$ instead of $\mathbf{a} = \mathbf{b}$ [15].

Robustness guarantees that the legitimate user can recover the secret from a hashed fuzzy vault. To understand what is needed to protect against the correlation attack consider the following example. The attacker is given two fuzzy vaults and is asked to guess weather they are made from the same fingerprint, or two different fingerprints. Let $M_v(v')$ be the guess of the attacker, where $M_v(v') = 1$ if he thinks that the two fuzzy vaults are made from the same fingerprint, and $M_v(v') = 0$ if he thinks that they are made from different fingerprints. We say that the *advantage* of the attacker in matching the vaults is $|P[M_v(v') = 1] - P[M_v(w) = 1]|$, where w is a random fuzzy vault. If the advantage is 0, then the attacker cannot tell the matching vault from a random vault. If the advantage is 1, the attacher will always be able to recognize a match. In general, we do not restrict the attacker to binary match/no match functions, but to arbitrary functions (i.e. functions which gives the confidence of two vaults matching). In the general case, as a measure of success for the attacker, we use statistical distance instead of the simple advantage in the example above.

Definition 3 (Δ Non-Linkability) *Let $v, v' \in \mathbb{Z}_r^{2C}$ be two fuzzy-vaults created from the same fingerprint with corresponding hash-functions $h, h' \in H$, respectively, and let w be a random fuzzy-vault. Recall that $M_v(w)$ is a probabilistic function that aligns fuzzy vault w to vault v and returns the number of points matched. The family of hash-functions H is Δ* non-linkable *if, for all probabilistic functions $M_v : \mathbb{Z}_r^{2C} \times \mathbb{Z}_r^{2C} \to m$, where $m \in \{0, \ldots, C\}$, the statistical distance between the probability distributions $M_v(w)$ and $M_v(v')$ is less than Δ:*

$$\frac{1}{2}\sum_{m=0}^{C} |P[M_v(w) = m] - P[M_v(v') = m]| \le \Delta. \tag{3}$$

To prevent correlation attacks, a family of hash functions must be Δ non-linkable for some sufficiently small value of Δ.

When designing a hash function for fuzzy-vaults, definition 3 requires that equation 3 is satisfied for *all* possible matching algorithms. Proving this property for all possible matching algorithms may be infeasible. In this case, one may instead try to prove equation 3 for a "large enough" class of matching algorithms. In Section 4.2 we propose a hash function which satisfies equation 3 for a large class of matching algorithms. We also argue why this class is a sufficient approximation to the full definition of Δ non-linkability.

For non-trivial Δ, a necessary property of a Δ non-linkable hash function is that the expected number of pre-images of a hashed point is greater than 1—otherwise the unique pre-image of two hashed vaults could easily be matched. This justifies our use of the term "hash-function." Besides being necessary for non-linkability, hash functions have the additional benefit of giving protection of the original fingerprint. To further guarantee this protection we formally define *non-invertability* as follows.

Definition 4 (Non-Invertability) *For all $(x, y) \in \mathbb{Z}_{p^2}$ the pre-image of the hash value $h(x, y)$ must not be unique:*

$$\#\{(x', y') \| h(x', y') = h(x, y)\} \geq 2. \tag{4}$$

Intuitively, a hash function is non-invertible if, for any given hash value $v = h(x, y)$, there is no unique point in the pre-images of v.

4.2. A family hash functions for secure non-linkable fuzzy vaults

Projecting the coordinates of vault points to a residue class modulo a prime number is a very efficient operation, and as shown in the following section, it satisfies all the three requirements explained in Section 4.1. In order to increase the number of possible hash functions, instead of a single prime number, we use two prime numbers for taking modulo. We define a family of hash functions $H = \{h_{p_1,p_2} : \mathbb{Z}_q \rightarrow \mathbb{Z}_{p_1 \times p_2}\}_{p_1,p_2<\sqrt{q}}$ as

$$h_{p_1,p_2}(x, y) = \begin{pmatrix} x \\ y \end{pmatrix} \mathrm{mod}(p_1, p_2), \tag{5}$$

where we use the notation $(x, y)^T \mathrm{mod}(p_1, p_2) = (x \bmod p_1, y \bmod p_2)$.

We usually omit the subscript of the hash function, when primes p_1 and p_2 are clear from the context.

In this work we choose the primes p_1 and p_2 between $\sqrt{q}/2$ and $\sqrt{q}/3$, since primes smaller than $\sqrt{q}/3$ produce relatively high FRR values while primes larger than $\sqrt{q}/2$ may cause unique pre-images for hash value $h(x, y)$, as shown in Theorem 3, and therefore do not satisfy the non-invertability property.

Note that the hash function as it is described in Equation 5 has a potential problem: for all $x < p_1$ and $y < p_2$, $h(x, y) = (x, y)$. This may make the hash-function linkable. To prevent this problem the minutia points, i.e. genuine points, are rotated by a small random angle before applying the hash function as shown in Equation 6. The same random rotation is applied to all minutia points in the vault:

$$h(x_i, y_i) = \mathbf{R}(x, y)^T \mathrm{mod}(p_1, p_2), \tag{6}$$

where $\mathbf{R}$ is a 2×2 rotation matrix that rotates the points in the (x, y) plane to add a small distortion intentionally.

To prevent inversion, the rotation is not stored. Notice that the additional rotation is only necessary in the enrollment phase, since the hashed fingerprint provided in the verification step will not be stored. The distortion created by this rotation does not require any modification of the verification phase, since the matching algorithm already tries to mitigate rotations due to the measurement process.

Hashing chaff points together with the minutia points may introduce collusion between vault points since two points (chaff or genuine) that are initially far apart, may be very close after hashing which will effect the false accept and false reject rates. To reduce collusion, we first hash the genuine points and later add random chaff points according to the method proposed in [1] such that x coordinates of the chaff points are at distance at least t from $\overline{x}_i$-coordinates of any of n genuine points and they are at distance at least t' from each other where $t' > t$.

The block diagram of the proposed fuzzy vault enrolling and verification scheme is shown in Figure 3 and Figure 4.

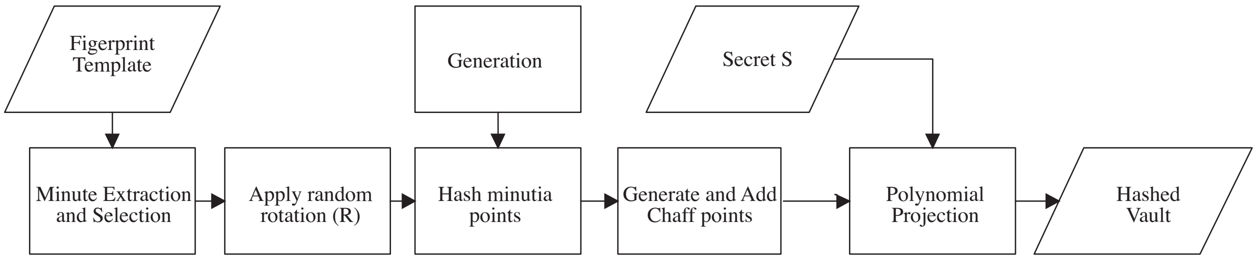

Figure 3. Proposed Vault Enrolling.

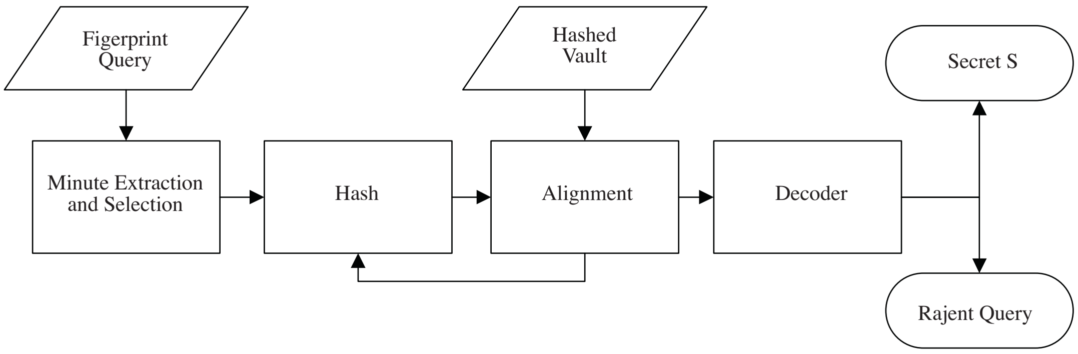

Figure 4. Proposed Vault Verification.

Note that the proposed verification is different from the original scheme only in the alignment phase, where the proposed alignment phase first hashes the query points and then aligns them with the vault. Recall that, in the alignment phase many rotations and translations are applied to the query vault to mitigate the noise between two measurements as described in Section 2.3. The details of the alignment phase are given in Algorithm 1.

Algorithm 1 Algorithm of the alignment phase.

```
1: for all rotations and translations
2:     Apply the rotation and translation to the query minutia points
3:     Hash them with the public hash function
4:     Compare the distance with Hashed Vault points
5:     if #matched points is larger than maximum matched
6:        Update maximum matched and keep this set of points
7:     end if
8: end for
```

There are two very fundamental differences between password driven transformation methods and our hash driven work in terms of utility and security. From the utility point of view, different from a password based system [7], user does not need to remember or store a secret in our scheme. The hash function is randomly generated during the enrollment phase and rotation used in the hash function is deleted after the hashing is applied; it is not stored in any way. From the security point of view, if the password is compromised, the fingerprint is also compromised. And it is very well known that the passwords do not provide adequate level of security. However, in our proposed scheme, the parameters (the prime values used for modulo operation) are public and no analysis can compromise the fingerprint. Moreover, storing password on a server compromises security if the server is compromised.

4.3. Property analysis of the proposed hash function

A hash function should satisfy the three properties given in Section 4.1. We claim that the hash function we proposed in Section 4.2 satisfies these properties.

Theorem 1 (Robustness) *The proposed family of hash functions is robust (Definition 2).*

Proof See Appendix A.1 for proof. □

Ideally speaking, the proof for Δ non-linkability (Definition 3) has to work for all probabilistic functions $M_v : \mathbb{Z}_r^{2C} \times \mathbb{Z}_r^{2C} \to m$. However, the following proof only covers minutia location based matchers such as the one used in this work (we henceforth refer to this class of matchers as matchers'). Nevertheless, we are confident that this is sufficient for the security of the scheme since all of the previous works on fingerprint verification, use matchers either based on similarity between coordinates of minutia points [6] or similarity between relative positions (angle and distance) of minutia points [16] where both of them are minutia location based matchers.

Before giving the proof of Δ non-linkability, we give two lemmas for the probability mass functions of M_v: one for two random vaults, and one for two vaults that are created from the same biometric data. These probability functions are necessary for the proof of Δ non-linkability.

For two hashed vaults made from distinct fingerprints (i.e. two random vaults), the probability that a point from the query vault matches to a point in the enrolled vault is denoted p_c where p_1, p_2 and p'_1, p'_2 are the parameters of the hash functions for enrolled and query vaults respectively. We assume that matching of two points are independent events since the point locations of vaults made from different fingerprints are mutually independent and behave randomly.

Lemma 1 *Let two fuzzy vaults, v, w, made from distinct fingerprints be given, and let $M_v(w)$ be the number of matching points after applying a matcher, then*

$$P[M_v(w) = x] = f(x; C, p_c) = \binom{C}{x} (p_c)^x (1 - p_c)^{C-x}, \tag{7}$$

for $x \geq 0$ and $p_c = \frac{C(2\delta)^2}{\max(p_1 p_2, p_1' p_2')}$.

Proof See Appendix A.2 for proof. □

Lemma 2 *Let two fuzzy vaults, v, v', made from identical fingerprints be given. Let $M_v(v')$ be the number of matching points after applying a matcher and let A be the total minutia point area before hashing (e.g. q), then*

$$P[M_v(v') = x] = f(x; n, p_g) = \binom{C}{x} (p_g)^x (1 - p_g)^{C-x},$$

where p_g is the probability that a point from the query vault matches a point in the enrolled vault, when the two hashed vaults are made from different measurements of the same fingerprint. Let us assume the worst case where both hash functions use the same prime values in hashing (i.e. $p_1 = p_1', p_2 = p_2'$):

$$p_g = p_c \left(1 - \frac{p_1 p_2 n}{AC}\right) + \frac{p_1 p_2 n}{AC}.$$

Proof See Appendix A.3 for proof. □

Theorem 2 (Δ Non-Linkability) *Assuming that matchers are the best possible matchers, the proposed family of hash-functions is Δ Non-Linkable (Definition 3), for*

$$\Delta = \frac{p_1 p_2 n (1 - p_c)^{C - \overline{x}} p_c^{\overline{x}}}{AC \int_0^1 t^{C - \overline{x} - 1} (1 - t)^{\overline{x}} dt}, \tag{8}$$

where p_1, p_2, p_c, p_g, n, A and C are as defined in Lemma 2 and $\overline{x}$ is the number which satisfies $(p_c)^{\overline{x}} (1 - p_c)^{C - \overline{x}} = (p_g)^{\overline{x}} (1 - p_g)^{C - \overline{x}}$.

Proof See Appendix A.4 for proof. □

Section 4.4 below, shows that Δ is small for the values of the parameters used in real life applications.

Theorem 3 (Non-Invertability) *The proposed family of hash-functions is non-invertible (Definition 4).*

Proof Recall that

$$h(x, y) = \mathbf{R} \begin{bmatrix} x \\ y \end{bmatrix} \bmod(p_1, p_2) \tag{9}$$

Due to the modulus operation, for all y there exist at least $\sqrt{q}/p_1 \geq 2$ possible pre-images for x and similarly $\sqrt{q}/p_2 \geq 2$ possible pre-images y for fixed x. □

4.4. Security against correlation attack

The correlation attack suggested by Scheirer et al. [5] depends on identifying the genuine points that are common in both vaults by matching the $\overline{x}_i$ values in the two vaults as defined in Section 2.1. The matching of the $\overline{x}_i$ values in the two vaults is done by exhaustive search [6] where the matching algorithm tries many possible rotations and translations and chooses the one that maximizes the number of matching points in the two vaults.

In this section we address linking: given a set of fuzzy vaults and a fuzzy vault made from a fingerprint which is present in the set, can an attacker find the matching fuzzy vaults? This is the attack by Scheirer et al. [5].

Theorem 4 (Correlation Resilience) *Let H be a family of hash-functions satisfying Definition 3. Furthermore, let an attacker have access to a fuzzy vault v, and a database of N vaults where exactly one of them is created from the same biometric information with v, and where all fuzzy vaults are created with the family of hash-functions H. The probability that the attacker correctly identifies the matching vault is*

$$P[right] \leq \frac{1}{N-1} + \psi.$$

Proof See Appendix A.5 for proof. □

Assume that an attacker captures two *fuzzy vaults* (v, v') that are created by using the same biometric data but with different hash functions. Theorem 2 shows that it is not possible to link the two hashed vaults created by our proposed hash function from the same biometric data but with different prime values, with a confidence greater than Δ. For the vault size $C = 250$, threshold $\delta = 3$ and primes 211 and 199 which is an optimal case according to our tests, our theoretical Equation 10 gives $p_c = 0.214$. For these values of C and p_c, $\epsilon = p_g - p_c$ is 0.013. With these parameters, $\Delta = 0.15$ as defined in Theorem 2.

Let N be 2000 and vault size be 250, which are reasonable values for a real life application. The probability of correctly guessing the matching vault with random guess is 0.0005 and the same probability with the best possible linking-algorithm is less than 0.001. Since the confidence is sufficiently small and there is no significant difference between a random guess and using the best possible linking algorithm, the vaults hashed with the proposed hash function are secure against the correlation attack.

5. Test results

We implement polynomial reconstruction phase with both of two previously discussed approaches: 1) brute force method and 2) RS decoding.

For the tests we use FVC2002-DB2 and our own database. FVC2002-DB2 is a public domain database with 100 fingers and 8 fingerprint images for each finger. Only the first two impressions of each finger were used in our experiments; the first impression was used as the template to encode the vault and the second impression was used as the query in vault decoding. Our own database has 180 fingers where there are two fingerprint images for each finger, for a total of 360 fingerprints. Similarly, the first 180 fingerprint images are used for enrollment and the second 180 images are used for verification of the corresponding fingerprints. Later, all fingerprints are cross-tested for false accept rates. In the experimental setting bitmap images of 500 × 500 pixels are created for each fingerprint. We set w to 9 bits and the secret polynomial is defined in $\mathbb{Z}_q[x]$ where $q = 262147$ which is a $2w$ bit prime. The codes are developed in either MATLAB or C++ (Microsoft Visual Studio), depending on the nature of the problem.

The alignment algorithm we use (Algorithm 1) does 12 translations in both x and y coordinates in range -25 to $+25$ pixels from the origin of the fingerprint and also does 100 rotations around the origin of the fingerprint. Note that since the number of rotations and translations are constant, the alignment algorithm is a $O(n)$ time algorithm where n is the number of minutia points in the query fingerprint.

We investigate mainly two issues; firstly, the effects of the vault sizes on the performance and security of the *fuzzy vault* for cases with and without using hash functions, and secondly, matcher results demonstrating that the correlation attack does not work when the proposed hash function is used.

5.1. Effects of vault sizes and usage of hash function

The false reject rates (FRR) and false accept rates (FAR) are calculated in four settings where different values for vault size are used for our database of fingerprints. We use vault sizes of 250 and 300 points and fix the degree of the polynomial used in vault encoding as 9. The minimum distance threshold between a genuine and a chaff point (t) is fixed as 18 and minimum distance between any two chaff points is taken as 8. For the hash function, we use primes 211 and 239 for the random modulus couples ($p = (p_1, p_2)$). Since the primes are both 8 bit values, q is the smallest 16 bit prime which is $q = 65537$. We calculate the FAR and FRR results for both with hashing and without hashing.

The FAR rates is 0% in all settings after cross testing all fingerprint images with different fingers in both databases.

Table 1. FRR of our work for different databases.

	Vault Size	
Hashing	250	300
Not Used	2.78%	3.89%
Used	5.55%	7.22%

(a) FRR for our Database

	Vault Size	
Hashing	250	300
Not Used	10%	14%
Used	13%	17%

(b) FRR for FVC2002-DB2

Table 2. FRR for FVC2002-DB2 in [7]

	Polynomial degree	
Password	8	10
Not Used	9%	14%
Used	12%	19%

Table 1(a) and Table 1(b) shows the FRRs for different vault sizes and usage of hash function and Table 2 shows the FRRs in the work of Nandakumar et al. [7] for the same database as in Table 1.2(b). The results clearly demonstrate that as more chaff points are added to the vault, the possibility of a genuine point matching to a chaff point increases resulting in higher FRRs. However, larger vault size results in higher security against brute force attack. The security impact of vault size was analyzed in [1]. Similarly, both for passwords [7] and hash functions, when they are used, FRRs increases in a very similar way.

Another issue that requires investigation is the effect of hash usage in the timing of the verification stage which determines the practicality of the system. The specific family of hash function family used in this work allows fast computation of hash values of vault points since only single precision arithmetic operations are needed, which are computed in a single clock cycle on vast majority of microprocessors. Furthermore, the verification stage timing is dominated by the alignment stage in comparison with which hash computation timing is negligible. To confirm this claim, we run the verification stage for 30 different fingerprints with two different sets of hash parameters and found out that hash usage increases the verification timing by about 10% on average. In addition, the experiment reveals that the maximum increase in the timing is 13.3% while the minimum is only 6.1%. The verification process can be performed sufficiently fast on either hardware or software platforms, and therefore 10% increase in verification timing is negligible from the point of practical usage.

5.2. Matcher results for correlation attack

As proven in Lemma 2, the expected number of points matched for two random vaults is almost equal to the number of points matched for two vaults created from the same fingerprint with different hash functions. The histograms of the number of points matched for both random vaults and matching vaults are presented in Figure 5 and Figure 6, where the vault size is 250 and primes are 211 and 199.

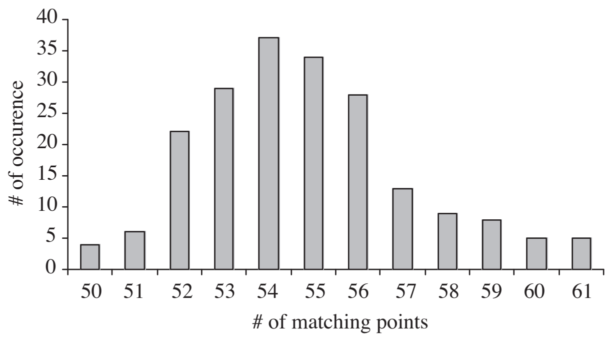

Figure 5. Histogram for random vaults.

The histograms clearly shows that it is not possible to link two vaults with more than a very small confidence since the distribution of the number of points is almost the same for matching vaults and random vaults.

For the parameters used in this tests, our theoretical equation 10 gives $p_c = 0.214$ and $p_g =$ is 0.227. Our theoretical results show that mean value of the number of points matched when both vaults are chosen randomly is $Cp_c = 53.5$, which complies with our empirical tests shown in Figure 5. Similarly for the case when two matching vaults are used, theoretical mean value is $Cp_g = 56.7$, which also complies with our empirical tests shown in Figure 6.

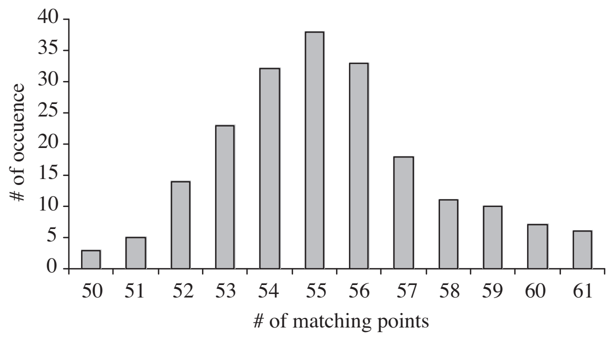

Figure 6. Histogram for Matching Vaults.

6. Conclusion

In this work, we addressed several issues of the fuzzy vault scheme for biometric identification such as security, performance, privacy and usability. We first provided a detailed guideline that explains the implementation steps of secure and efficient fuzzy vaults for the particular case of fingerprints, using two methods: brute force and Reed-Solomon (RS).

We proposed a modification to the fuzzy vault scheme in which a family of special hash functions is used to preserve the distances between points in the fuzzy vault. According to the scheme, points in the vault (both genuine and chaff points) are first hashed and then stored; a subsequent verification performs the matching in the hash domain. Since the proposed hash function is not invertible, it is computationally infeasible to extract the biometric and the secret polynomial embedded in the vault without presenting the biometric first.

We defined the properties that a family of hash functions should satisfy and gave intuitive explanations of these properties. We then proposed a family of hash functions, and gave formal proofs that it satisfies these definitions. We also presented the security analysis of the modified scheme which showed that the correlation attack is inapplicable if a family of hash functions which satisfies the proposed definitions is used.

We implemented the proposed scheme for fingerprint biometric identification to investigate the performance implications of the scheme on false accept and false reject rates. We observed that biometric identification can efficiently be performed using hashed vault points and that the false accept rate is 0 for all cases. The proposed scheme is observed to incur a slight degradation in false reject rate which is acceptable in identification systems and less than other implementations in the literature.

While larger fuzzy vaults (i.e. those with more chaff points) improve the overall security, the adverse affect of more chaff points on the false reject rate can be observed on the conventional fuzzy vault scheme as well as on its hashed version. However, the brute force attack in [9] whose complexity depends on vault size can recover only the hashed minutia points, and therefore the biometric remains secure even after the brute force attack. Therefore, our framework provides additional security against the brute force attack as well without a need for a very large vault.

As better algorithms for the verification stage, or alternative families of hash functions, can potentially improve the false reject rate, we are confident that the proposed framework will be instrumental in making use of the fuzzy vault scheme in a more secure way.

Acknowledgments

This work is partially supported by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under project number 105E165. We would like to thank TÜBİTAK for the PhD. fellowship supports granted to Cengiz Örencik. We also thank Berrin Yanıkoğlu and Alisher Kholmatov for bringing the problem to our attention and their valuable support.

References

[1] C. Orencik, T. B. Pedersen, E. Savas and M. Keskinoz, Improved Fuzzy Vault Scheme for Fingerprint Verification, in *ICETE 2008, International Joint Conference on e-Business and Telecommunications SECRYPT.*

[2] Juels and M. Sudan, Fuzzy Vault Scheme, in *IEEE International Symposium on Information Theory*, p 408, 2002.

[3] C. Clancy, N. Kiyavash and D. Lin, Secure Smartcard - Based Fingerprint Authentication, in *ACM Workshop on biometric methods and applications, (WBMA)*, 2003.

[4] U. Uludag, S. Pankanti and A. Jain, Fuzzy Vault for Fingerprints, in *Proceeding of International Conference on Audio- and Video-Based Biometric Person Authentication*, pp. 310-319, 2005.

[5] W. J. Scheirer and T. E. Boult, Cracking Fuzzy Vaults and Biometric Encryption, in *IEEE Biometrics Research Symposium at the National Biometrics Consortium Conference*, 2007.

[6] A. Kholmatov and B. Yanikoglu, Realization of Correlation Attack Against Fuzzy Vault, in *Security, Forensics, Steganography and Watermarking of Multimedia Contents X, Electronic Imaging*, 2008.

[7] K. Nandakumar, A. Nagar and A. K. Jain, Hardening Fingerprint Fuzzy Vault using Password, in *International Conference on Biometrics*, pp. 927-937, 2007.

[8] Ron M. Roth, *Introduction to Coding Theory*, Cambridge University Press, 2006.

[9] P. Mihailescu, The fuzzy vault for fingerprints is vulnerable to brute force attack, http://arxiv.org/abs/0708.2974v1, 2007.

[10] Kenneth H. Rosen, *Elementary Number Theory and its applications*, Pearson Addison Wesley, 2005.

[11] V. Guruswami and M. Sudan, Improved Decoding of Reed-Solomon and Algebraic-Geometric codes, in *FOCS 1998, Symposium on Foundations of Computer Science*, 1998.

[12] M. Sudan, Decoding of Reed Solomon Codes Beyond the Error Correction Bound, in *Journal of Complexity*, pp. 180-193, 1997.

[13] Qiong Li, Zhaoqing Liu and Xiamu Niu, Analysis and Problems on Fuzzy Vault Scheme, in *International Conference on Intelligent Information Hiding and Multimedia*, pp. 244-250, 2006.

[14] K. E. Atkinson, *An Introduction to Numerical Analysis*, John Willey & Sons, 1988.

[15] Wikipedia, Metric (mathematics) — Wikipedia, The Free Encyclopedia, 2009.

[16] S. Yang and I. Verbauwhede, Automatic Secure Fingerprint Verification System based on Fuzzy Vault Scheme, in *Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 609-612, 2005.

[17] K. Pearson, *Tables of Incomplete Beta Functions, 2nd ed.*, Cambridge University Press, 1968.

APPENDIX

A. Proofs

A.1. Proof of Theorem 1:

Proof Assume (x, y) and (x', y') are two points where the Chebyshev distance between them is less than δ. This implies that $|x - x'| < \delta$ and $|y - y'| < \delta$. In particular, $\delta > |x - x'| \geq \min(|x \bmod p - x' \bmod p|, p - |x \bmod p - x' \bmod p|) = |x - x'|_p$, and likewise $\delta > |y - y'|_p$. This gives the desired result:

$$\|h(x, y) - h(x', y')\|_\infty = \max(|x - x'|_p, |y - y'|_p) < \delta$$

□

A.2. Proof of Lemma 1

Proof Recall that C is the total number of points in both vaults, and two points, one from each vault, is called a match if the Chebyshev distance between them is less than δ. As illustrated in Figure 7, the vault points are distributed in an area of size $\max(p_1p_2, p'_1p'_2)$ and a query point matches to a vault point if it is within a $2\delta \times 2\delta$ area centered at the vault point. The probability that a random point in the query vault match to a point in the enrolled vault is:

$$p_c = \frac{C(2\delta)^2}{\max(p_1p_2, p'_1p'_2)}. \tag{10}$$

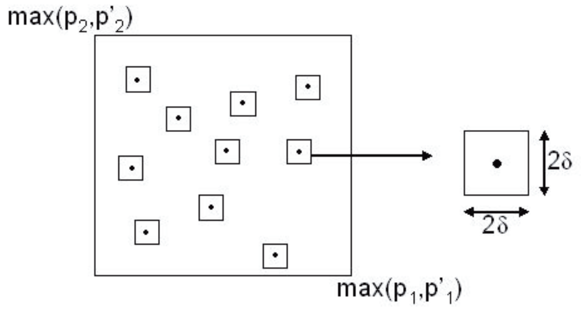

Figure 7. Hashed vault area and matching range.

The number of matching points has binomial distribution, since we assume that the matching of points are independent events. Therefore, $P[M_v(w) = x]$ can be calculated by the following equation:

$$P[M_v(w) = x] = \binom{C}{x} (p_c)^x (1 - p_c)^{C-x},$$

for $x \geq 0$. □

A.3. Proof of Lemma 2

Proof Recall that n is the number of genuine points in a vault, and that the hashed vault is created by a random rotation followed by a modulo operation:

$$(x'_i, y'_i) = h(x_i, y_i) = \mathbf{R}(x_i, y_i)^T \mathrm{mod}(p_1, p_2).$$

In the alignment phase, the matcher finds the rotation $\mathbf{R}$ and the translation T, which gives the maximum number of overlapping points in the two hashed vaults. Since p_1 and p_2 are in the range $[\sqrt{q}/3, \sqrt{q}/2]$, given a minutia point (x, y) there are four equally likely cases for the rotation and translation which will map the hashed minutia point $h(x, y)$ back to the original point (x, y), where $[\mathbf{R}(x_i, y_i)^T]_x$ is the x coordinate of $\mathbf{R}(x_i, y_i)^T$ and similarly for y coordinate:

1. $0 \leq [\mathbf{R}(x_i, y_i)^T]_x < p_1$ and $0 \leq [\mathbf{R}(x_i, y_i)^T]_y < p_2$
   $\Rightarrow (x_i, y_i)^T = \mathbf{R}^{-1}(x'_i, y'_i)^T$.

2. $p_1 \leq [\mathbf{R}(x_i, y_i)^T]_x < 2p_1$ and $0 \leq [\mathbf{R}(x_i, y_i)^T]_y < p_2$
   $\Rightarrow (x_i, y_i)^T = \mathbf{R}^{-1}(x'_i, y'_i)^T + \mathbf{R}^{-1}(p_1, 0)^T$.

3. $0 \leq [\mathbf{R}(x_i, y_i)^T]_x < p_1$ and $p_2 \leq [\mathbf{R}(x_i, y_i)^T]_y < 2p_2$
   $\Rightarrow (x_i, y_i)^T = \mathbf{R}^{-1}(x'_i, y'_i)^T + \mathbf{R}^{-1}(0, p_2)^T$.

4. $p_1 \leq [\mathbf{R}(x_i, y_i)^T]_x < 2p_1$ and $p_2 \leq [\mathbf{R}(x_i, y_i)^T]_y < 2p_2$
   $\Rightarrow (x_i, y_i)^T = \mathbf{R}^{-1}(x'_i, y'_i)^T + \mathbf{R}^{-1}(p_1, p_2)^T$.

As illustrated in Figure 8, there are other less probable cases, but since the matcher tries a large number of rotations and translations and chooses the rotation and translation, which maximizes the number of points matched, we assumed the worst case (best case for attacker) that matcher finds the rotation which falls in one of the most possible four cases above.

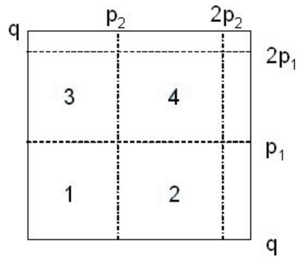

Figure 8. Vault area before hashing and matching cases.

Let the best rotation $\mathbf{R}$, and translation T be given. We divide the matching points into two sets: the matching points which are "real matches" (the two points correspond to the same minutia point), and the set of points which are "false matches" (either one of the points is a chaff point, or they are both genuine points, but correspond to different minutia points).

Notice that, to get a *real match*, the chosen rotation and translation will fall in one of the four cases above. Assume the matcher finds the inverse rotation and translation for one of the four cases above (say j^{th} case), by first applying this rotation and translation, and then taking the modulo of a query point from the j^{th} case, it matches with the enrolled minutia point. However if the query point is not from the j^{th} case, it requires a different rotation translation pair than the one applied, so getting a *real match* is not possible.

Under the assumption that the vault points are uniformly distributed, only $\frac{p_1 p_2}{A}$ of the points can be under the same case. Therefore, for any rotation and translation only $\frac{p_1 p_2}{A}$ of the points can match and the rest of the genuine points behave like chaff points.

Let $P[\text{chaff}]$ denote the probability of a query vault point being a chaff point, $P[\text{match\_c}]$ be the probability that a chaff query point match to a point, $P[\text{genuine}]$ denote the probability of a query vault point being a genuine point, $P[\text{match\_g}]$ be the probability that a genuine query point match to a point. Note that it is not important whether the matched point is genuine or chaff. The total probability that a point from the query vault match to a point in the enrolled vault is:

$$\begin{aligned} p_g &= P[\text{chaff}]P[\text{match\_c}] + P[\text{genuine}]P[\text{match\_g}] \\ &= \frac{C-n}{C}p_c + \frac{n}{C}\left(\frac{p_1p_2}{A} + \left(1 - \frac{p_1p_2}{A}\right)p_c\right) \\ &= p_c\left(1 - \frac{n}{C} + \frac{n}{C} - \frac{np_1p_2}{AC}\right) + \frac{np_1p_2}{AC} \\ &= p_c\left(1 - \frac{np_1p_2}{AC}\right) + \frac{np_1p_2}{AC}, \end{aligned}$$

$$P[M_v(v') = x] = f(x; C, p_g) = \binom{C}{x} (p_g)^x (1-p_g)^{C-x}.$$

□

A.4. Proof of Theorem 2

Proof Let v, v' be fuzzy vaults created from identical fingerprints, and let w be a fuzzy vault created from a different fingerprint. The two probability distributions $(M_v(w))$ and $(M_v(v'))$ represents the number of matched points obtained when a matcher is used to match w and v' respectively to v.

Recall that due to Lemma 2, the probability that a point from v' matches a point from v is

$$p_g = p_c\left(1 - \frac{p_1p_2n}{AC}\right) + \frac{p_1p_2n}{AC}.$$

Let

$$\epsilon = \frac{p_1p_2n}{AC}(1-p_c),$$

then

$$p_g = p_c + \epsilon.$$

Let T be a constant integer, $0 \leq p < 0.5$, then

$$(p)^x (1-p)^{T-x} = p^T \left(\frac{1-p}{p}\right)^{T-x}$$

is a monotonically decreasing function for x in the range $x = 0$ to $x = T$. Let f_1 and f_2 be two monotonically decreasing functions in the range $x = 0$ to $x = T$ where $f_1(0) > f_2(0)$ and $f_1(T) < f_2(T)$. Then there is a unique value $\overline{x}$ where $f_1(\overline{x}) = f_2(\overline{x})$.

Under the assumption that $p_c, p_c + \epsilon < 0.5$, there exists a unique value $\overline{x}$ where

$$(p_c)^{\overline{x}} (1-p_c)^{C-\overline{x}} = (p_c+\epsilon)^{\overline{x}} (1-p_c-\epsilon)^{C-\overline{x}} \tag{11}$$

and

$$\begin{array}{l} P[M_v(v') = x] \leq P[M_v(w) = x], \text{ for } 0 \leq x \leq \overline{x}, \\ P[M_v(v') = x] > P[M_v(w) = x], \text{ for } \overline{x} < x \leq C. \end{array}$$

By splitting the sum of the statistical distance into these two cases, we get

$$\begin{aligned} & \frac{1}{2}\left(\sum_{x=0}^{C} |P[M_v(v') = x] - P[M_v(w) = x]|\right) \\ = & \frac{1}{2}((P[M_v(w) \leq \overline{x}] - P[M_v(v') \leq \overline{x}])) + \\ & \frac{1}{2}((P[M_v(v') > \overline{x}] - P[M_v(w) > \overline{x}])) \\ = & \frac{1}{2}(P[M_v(w) \leq \overline{x}] - P[M_v(v') \leq \overline{x}]) + \\ & \frac{1}{2}((1 - P[M_v(v') \leq \overline{x}]) - (1 - P[M_v(w) \leq \overline{x}])) \\ = & \frac{1}{2}2(P[M_v(w) \leq \overline{x}] - P[M_v(v') \leq \overline{x}]). \end{aligned} \tag{12}$$

From [17] we get that, for a binomially distributed random variable X,

$$P[X \leq k] = I_{1-p}(C-k, k+1) = \frac{\int_0^{1-p} t^{C-k-1}(1-t)^k dt}{\int_0^1 t^{C-k-1}(1-t)^k dt}.$$

Using this fact in Equation 12 gives us the relation

$$\begin{aligned} & \frac{1}{2}\left(\sum_{x=0}^{C} |P[M_v(v') = x] - P[M_v(w) = x]|\right) \\ = & (P[M_v(w) \leq \overline{x}] - P[M_v(v') \leq \overline{x}]) \\ = & I_{1-p_c}(C-\overline{x}, \overline{x}+1) - I_{1-p_c-\epsilon}(C-\overline{x}, \overline{x}+1) \\ = & \frac{\int_{1-p_c-\epsilon}^{1-p_c} t^{C-\overline{x}-1}(1-t)^{\overline{x}} dt}{\int_0^1 t^{C-\overline{x}-1}(1-t)^{\overline{x}} dt} \end{aligned} \tag{13}$$

Note that, since the derivative of $(t^{C-\overline{x}-1}(1-t)^{\overline{x}})$ is positive,

$$\begin{aligned}\frac{\partial}{\partial t}(t^{C-\overline{x}-1}(1-t)^{\overline{x}}) &= (C-\overline{x}-1)t^{C-\overline{x}-2}(1-t)^{\overline{x}} \\ &\quad -\overline{x}t^{C-\overline{x}-1}(1-t)^{\overline{x}-1} > 0;\end{aligned}$$

for $1-p_c-\epsilon \leq t \leq 1-p_c$, the term $(t^{C-\overline{x}-1}(1-t)^{\overline{x}})$ is monotonically increasing. We can thus upper bound the integral with the interval length, ϵ, times the largest value: $\int_{1-p_c-\epsilon}^{1-p_c} t^{C-\overline{x}-1}(1-t)^{\overline{x}}dt \leq \epsilon(1-p_c)^{C-\overline{x}-1}p_c^{\overline{x}}$. Thus

$$\begin{aligned}&\frac{1}{2}\left(\sum_{x=0}^{C}|P[M_v(v')=x]-P[M_v(w)=x]|\right) \\ &\qquad \leq \frac{\epsilon(1-p_c)^{C-\overline{x}-1}p_c^{\overline{x}}}{\int_0^1 t^{C-\overline{x}-1}(1-t)^{\overline{x}}dt}. \qquad (14)\end{aligned}$$

Setting $\Delta = \frac{\epsilon(1-p_c)^{C-\overline{x}-1}p_c^{\overline{x}}}{\int_0^1 t^{C-\overline{x}-1}(1-t)^{\overline{x}}dt}$, completes the proof. $\square$

A.5. Proof of Theorem 4:

Proof Let v' denote the vault in the database which is created from the same fingerprint as v. Furthermore, let match(x,y) be the best possible function an attacker can use to verify if the two vaults x and y match (match$(x,y)=1$ if the attacker thinks that x and y match, and 0 otherwise). With such a function the attacker tries to match each of the N vaults in the database to the given fuzzy vault v. Of all the vaults that match, the best the attacker can do is to pick one vault at random, and return it as his guess for the matching vault. Note that, since function match(x,y) satisfies Definition 3,

$$\frac{1}{2}\sum_{i=0}^{1}|P[\text{match}(v,V')=i]-P[\text{match}(v,W)=i]| \leq \Delta$$

We assume that

$$P[\text{match}(v,V')=1] \geq 0.5 \geq P[\text{match}(v,W)=1], \qquad (15)$$

where V is a random variable representing fuzzy vaults created from the same fingerprint as the vault v, and W represents random fuzzy vaults. If the assumption does not hold, then the function *match* performs worse then a random guess.

Let 'right' denote the event of correctly guessing the matching vault, 'In_Set' denote the event that the matching vault is one of the vaults which pass the matching algorithm (e.g. the event match$(v,v')=1$) and 'pick_right' denote the event that the attacker picks the correct vault from the reduced set. Then

$$P[\text{right}] = P[\text{In\_Set}]P[\text{pick\_right}|\text{In\_Set}]. \qquad (16)$$

From Definition 3 we have that $\Delta \geq |P[\text{match}(v,V')=1]-P[\text{match}(v,W)=1]|$, which, by Equation 15, gives us $P[\text{match}(v,V')=1] \leq \Delta + P[\text{match}(v,W)=1]$. Note that the event 'In_Set' is exactly the event that

$\text{match}(v, V') = 1$, so

$$P[\text{In\_Set}] \leq (P[\text{match}(v, W) = 1] + \Delta) \leq 0.5 + \Delta. \tag{17}$$

Putting Definition 3 and Equation 15 together also tells us that there are an expected $(P[\text{match}(v, V') = 1] - \Delta)(N - 1) \geq (0.5 - \Delta)(N - 1)$ vaults wrongly selected as a possible candidate by the matching algorithm. For the attacker, each of the elements in the candidate list is equally probable to be the correct vault, so $P[\text{pick\_right}|\text{In\_Set}] \leq 1/((0.5 - \Delta)(N - 1) + 1)$. Therefore,

$$\begin{aligned}
P[\text{right}] &= P[\text{In\_Set}]P[\text{pick\_right}|\text{In\_Set}] \\
&\leq \frac{(0.5 + \Delta)}{(0.5 - \Delta)(N - 1) + 1} \\
&= \frac{(0.5 - \Delta) + 2\Delta}{(0.5 - \Delta)(N - 1) + 1} \\
&< \frac{(0.5 - \Delta)}{(0.5 - \Delta)(N - 1)} + \frac{2\Delta}{(0.5 - \Delta)(N - 1) + 1} \\
&= \frac{1}{(N - 1)} + \frac{2\Delta}{(0.5 - \Delta)(N - 1) + 1}.
\end{aligned}$$

Setting $\psi = \frac{2\Delta}{(0.5-\Delta)(N-1)+1}$ completes the proof. $\square$