Fuzzy Vault for Fingerprints

Umut Uludag[1], Sharath Pankanti[2], and Anil K. Jain[1]

[1] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824
{uludagum,jain}@cse.msu.edu
[2] Exploratory Computer Vision Group, IBM T.J. Watson Research Center, Yorktown Heights, NY, 10598
sharat@us.ibm.com

Abstract. Biometrics-based user authentication has several advantages over traditional password-based systems for standalone authentication applications, such as secure cellular phone access. This is also true for new authentication architectures known as *crypto-biometric* systems, where cryptography and biometrics are merged to achieve high security and user convenience at the same time. In this paper, we explore the realization of a previously proposed cryptographic construct, called *fuzzy vault*, with the fingerprint minutiae data. This construct aims to secure critical data (e.g., secret encryption key) with the fingerprint data in a way that only the authorized user can access the secret by providing the valid fingerprint. The results show that 128-bit AES keys can be secured with fingerprint minutiae data using the proposed system.

1 Introduction

In traditional cryptography one or more keys are used to convert the plain text (data to be encrypted) to cipher text (encrypted data): the encrypting key(s) maps the plain text to essentially a sequence of random bits, that can only be mapped back to the plain text using the appropriate decrypting key(s). Without the knowledge of the correct decrypting keys, the conversion of cipher text to the plain text is *infeasible* considering time and cost limitations [1]. Hence, the cipher text is *secured*: even if an attacker obtains the cipher text, she cannot extract useful information from it. Here, the plain text can be any data that needs to be stored or transmitted securely: financial transactions, email communication, secret cryptographic keys, etc. Current cryptographic algorithms (e.g., Advanced Encryption Standard (AES) [2], Data Encryption Standard (DES) [1], RSA [1]) have a very high proven security but they suffer from the key management problem: all these algorithms fully depend on the assumption that the keys will be kept in absolute secrecy. If the secret key is compromised, the security provided by them immediately falls apart. Another limitation of the these algorithms is that they require the keys to be very long and random for higher security, e.g., 128 bits for AES [2], which makes it impossible for users to memorize the keys. As a result, the cryptographic keys are stored securely (e.g., in a computer or on a smart card) and released based on some alternative authentication mechanism. If this authentication succeeds, keys can be used in encryption/decryption procedures.

The most popular authentication mechanism used for key release is based on passwords, which are again cryptographic key-like strings but simple enough for users to

T. Kanade, A. Jain, and N.K. Ratha (Eds.): AVBPA 2005, LNCS 3546, pp. 310–319, 2005.

users to remember. Hence, the plain text protected by a cryptographic algorithm is only as secure as the password (weakest link) that releases the correct decrypting keys. Simple passwords compromise security, but complex passwords are difficult to remember and expensive to maintain. Further, passwords are unable to provide non-repudiation: a subject may deny releasing the key using password authentication, claiming that her password was stolen. Many of these limitations of password-based key release can be eliminated by incorporating biometric authentication. Biometric authentication [3] refers to verifying individuals based on their physiological and behavioral traits. It is inherently more reliable than password-based authentication as biometric characteristics cannot be lost or forgotten. Further, biometric characteristics are difficult to copy, share, and distribute, and require the person being authenticated to be present at the time and point of authentication. Thus, biometrics-based authentication is a potential candidate to replace password-based authentication, either for providing complete authentication mechanism or for securing the traditional cryptographic keys.

A biometric system and a cryptographic system can be merged in one of the following two modes: (i) In biometrics-based key release, the biometric matching is decoupled from the cryptographic part. Biometric matching operates on the traditional biometric templates: if they match, cryptographic key is released from its secure location, e.g., a smart card or a server. Here, biometrics effectively acts as a wrapper mechanism in cryptographic domain. (ii) In biometrics-based key generation, biometrics and cryptography are merged together at a much deeper level. Biometric matching can effectively take place within cryptographic domain, hence there is no separate matching operation that can be attacked; positive biometric matching *extracts* the secret key from the conglomerate (key/biometric template) data. An example of the biometric-based key generation, called *fuzzy vault,* was proposed by Juels and Sudan [4]. This cryptographic construct, as explained in later sections, has the characteristics that make it suitable for applications that combine biometric authentication and cryptography: the advantages of cryptography (e.g., proven security) and fingerprint-based authentication (e.g., user convenience, non-repudiation) can be utilized in such systems.

2 Background

Tuyls et al. [5] assume that a noise-free template X of a biometric identifier is available at the enrollment time and use this to enroll a secret S to generate a helper data W. If each dimension of the multidimensional template is quantized at q resolution levels, the process of obtaining W is akin to finding residuals that must be added to X to fit to odd or even grid quantum depending upon whether the corresponding S bit is 0 or 1. At decryption time, the noise-prone biometric template Y is used to decrypt W to obtain a decrypted message S' which is approximately the same as S. In each dimension, the process of decryption guesses whether a particular bit of secret S is 0 or 1 depending upon whether the sum of Y and W resides in even or odd quantum of the corresponding dimension. It is hoped that the relatively few errors in S' can be corrected using error-correction techniques. The proposed technique assumes that the biometric representations are completely aligned and that noise in each dimension is relatively small compared to the quantization magnitude Q. Due to variability in the

biometric identifier, different W may be generated for the same message S. The authors prove that very little information is revealed from W by appropriately tuning the quantization scheme with respect to the measurement noise.

Juels and Sudan's fuzzy vault scheme [4] is an improvement upon the previous work by Juels and Wattenberg [6]. In [4], Alice can place a secret κ (e.g., secret encryption key) in a vault and lock (secure) it using an unordered set A. Here, unordered set means that the relative positions of set elements do not change the characteristics of the set: e.g., the set {-2, -1, 3} conveys the same information as {3, -1, -2}. Bob, using an unordered set B, can unlock the vault (access κ) only if B overlaps with A to a great extent. The procedure for constructing the fuzzy vault is as follows: First, Alice selects a polynomial p of variable x that encodes κ (e.g., by fixing the coefficients of p according to κ). She computes the polynomial projections, $p(A)$, for the elements of A. She adds some randomly generated chaff points that do not lie on p, to arrive at the final point set R. When Bob tries to learn κ (i.e., find p), he uses his own unordered set B. If B overlaps with A substantially, he will be able to locate many points in R that lie on p. Using error-correction coding (e.g., Reed-Solomon [7]), it is assumed that he can reconstruct p (and hence κ). A simple numerical example for this process is as follows: Assume Alice selects the polynomial $p(x) = x^2 - 3x + 1$, where the coefficients (1, -3, 1) encode her secret κ. If her unordered set is $A = \{-1, -2, 3, 2\}$, she will obtain the polynomial projections as $\{(A, p(A))\} = \{(-1,5),(-2,11),(3,1),(2,-1)\}$. To this set, she adds two chaff points $C = \{(0,2),(1,0)\}$ that do not line on p, to find the final point set $R = \{(-1,5),(-2,11),(3,1),(2,-1),(0,2),(1,0)\}$. Now, if Bob can separate at least 3 points from R that lie on p, he can reconstruct p, hence decode the secret represented as the polynomial coefficients (1, -3, 1). Otherwise, he will end up with incorrect p, and he will not be able to access the secret κ.

The security of this scheme is based on the infeasibility of the polynomial reconstruction problem (i.e., if Bob does not locate many points which lie on p, he can not feasibly find the parameters of p, hence he cannot access κ). The scheme can tolerate some differences between the entities (unordered sets A and B) that lock and unlock the vault, so Juels and Sudan named their scheme *fuzzy vault*. This fuzziness can come from the variability of biometric data: even though the same biometric entity (e.g., right index finger) is analyzed during different acquisitions, the extracted biometric data will vary due to acquisition characteristics (e.g., placement of the finger on the sensor), sensor noise, etc. On the other hand, in traditional cryptography, if the keys are not exactly the same, the decryption operation will produce useless random data. Note that since the fuzzy vault can work with unordered sets (common in biometric templates, including fingerprint minutiae data), it is a promising candidate for biometric cryptosystems. Having said this, the fuzzy vault scheme requires prealigned biometric templates. Namely, the biometric data at the time of enrollment (locking) must be properly aligned with biometric data at the time of verification (unlocking). This is a very difficult problem due to different types of distortion that can occur in biometric data acquisition. Further, the number of feasible operating

points (where the vault operates with negligible complexity, e.g., conveyed via the number of required access attempts to reveal the secret, for a genuine user and with considerable complexity for an imposter user) for the fuzzy vault is limited: for example, the flexibility of a traditional biometric matcher (e.g., obtained by changing the system decision threshold) is not present.

Clancy et al. [8] proposed a fingerprint vault based on the fuzzy vault of Juels and Sudan [4]. Using multiple minutiae location sets per finger (based on 5 impressions of a finger), they first find the canonical positions of minutia, and use these as the elements of the set A. They added the maximum number of chaff points to find R that locks κ. However, their system inherently assumes that fingerprints (the one that locks the vault and the one that tries to unlock it) are pre-aligned. This is not a realistic assumption for fingerprint-based authentication schemes. Clancy et al. [8] simulated the error-correction step without actually implementing it. They found that 69-bit security (for False Accept Rate (FAR)) could be achieved with a False Reject Rate (FRR) of 20-30%. Note that the cited security translates to $2^{-69} \approx 1.7 \cdot 10^{-21}$ FAR. Further, FRR value suggests that a genuine user may need to present her finger multiple times to unlock the vault.

The design of a fuzzy vault (without the actual implementation) using minutiae-based lines was given in [9]. A more detailed survey of biometric cryptosystems can be found in [10].

3 Proposed Method

In this section we present our implementation of the fuzzy vault, operating on the fingerprint minutiae features. These features are defined as abrupt changes in the regular ridge structure on the fingertip, characterized by either ending or bifurcation of the ridges. Typically, they are represented as (x, y, θ) triplets, denoting their row indices (x), column indices (y) and angle of the associated ridge, respectively. These features are shown in Fig. 1, where two fingerprint images obtained from the same finger at different times, with overlaid minutiae are shown. The minutiae are found using the algorithm outlined in [11]. Note that the variability in the number and position of minutiae is evident in the two images.

(a) (b)

Fig. 1. Fingerprint images with overlaid minutiae: (a) template image (28 minutiae), (b) query image (26 minutiae)

Fig. 2 shows the block diagram of the proposed fingerprint fuzzy vault system. Fig. 3 shows the variables used in the system pictorially: the polynomial in Fig. 3(a) encodes the secret. It is evaluated at both genuine and chaff points in Fig. 3(b). Finally, the vault is the union of genuine and chaff points.

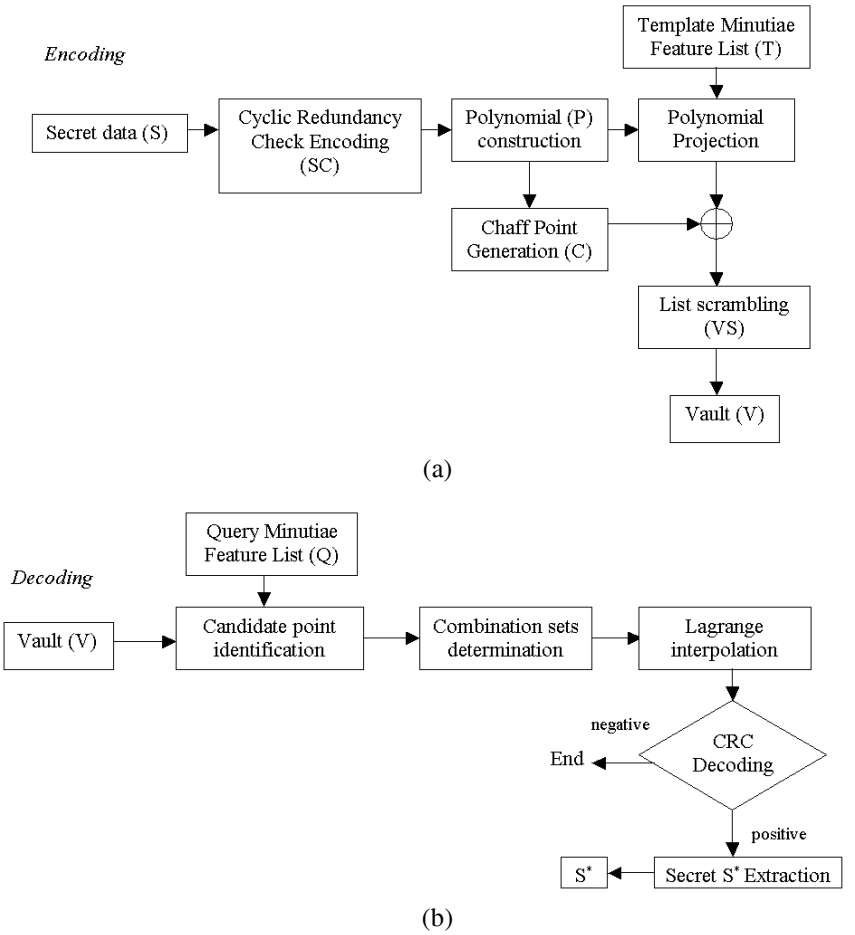

Fig. 2. Fuzzy fingerprint vault: (a) vault encoding, (b) vault decoding.

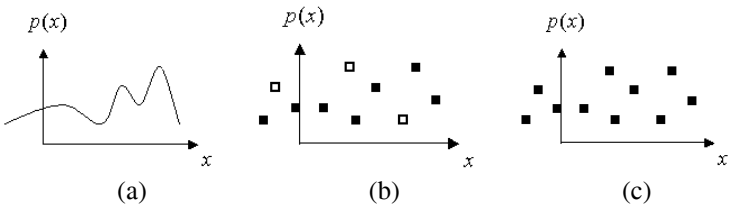

Fig. 3. System parameters: (a) polynomial, (b) evaluation of the polynomial (filled squares: genuine points, empty squares: chaff points), (c) final vault list

3.1 Encoding

Secret S is any data that needs to be protected, but the size of S that can be feasibly protected is limited by the capacity of the entity used for locking and unlocking the vault. Currently, we use x and y coordinates of minutiae points for locking/unlocking the vault. We first align the minutiae, namely compensate for the translation and rotation between template and query minutiae data. Encoding operation secures S with fingerprint minutiae data: if a query minutiae set similar to the template minutiae set is presented during decoding, it indicates the presence of an authorized person and S can be reconstructed accurately. Note that the vault operation is decoupled from any backend application (e.g., encryption/decryption using S): vault is only responsible for securing S with fingerprint data. The fingerprint template has the role of a key. Note that this is not the key in traditional cryptosystems (e.g., AES) per se: rather, it has the role of a key for a new cryptographic construct, namely the fuzzy vault. In the current implementation, S is generated as a 128-bit random bit stream. This can simulate securing AES symmetric encryption keys.

Our current decoding implementation does not include any error-correction scheme, as proposed by Juels and Sudan's [4], since there are serious difficulties to achieve error-correction with biometric data. Developing the necessary polynomial reconstruction via error-correction has not been demonstrated in the literature. Instead, our algorithm decodes many candidate secrets. To identify which one of these candidates is the actual secret, we need to put some structure into the secret S. By checking the validity of this structure during decoding, the algorithm can identify whether a given candidate secret is correct or not. Cyclic Redundancy Check (CRC) is a generalization of the simple parity bit checking. It is commonly used in communication channel applications for error detection where the errors are introduced due to channel noise. In our case using incorrect minutiae points during decoding will cause an incorrect polynomial reconstruction, resulting in errors. In the current implementation, we generate 16-bit CRC data from the secret S. Hence, the chance of a random error being undetected (i.e., failing to identify an incorrect decoding) is 2^{-16}. The 16-bit primitive polynomial, $g_{CRC}(a) = a^{16} + a^{15} + a^2 + 1$, we use for CRC generation is called "CRC-16" and is used for EBCDIC messages by IBM [12]. Appending the CRC bits to the original secret S (128-bits), we construct 144-bit data SC. From this point on, all operations take place in Galois fields with cardinality 65536, namely GF(2^{16}): we concatenate x and y coordinates of a minutiae (8-bits each) as $[x \mid y]$ to arrive at the 16-bit locking/unlocking data unit u. Note that to account for slight variations in minutiae data (due to nonlinear distortion), raw minutiae data are first quantized. Namely, each minutia is translated to lie in a square tessellation of the 2D image plane. For example, if the block size used in the tessellation is 7, any minutia that has x coordinate in the range [1, 7] is assumed to originate from x coordinate 4. This allows for ±3 pixel variations in the coordinates of template and query minutiae.

SC is used to find the coefficients of the polynomial p: 144-bit SC can be represented as a polynomial with 9 (144/16) coefficients in GF(2^{16}), with degree $D = 8$. Hence, $p(u) = c_8 u^8 + c_7 u^7 + \ldots + c_1 u + c_0$. Simply, SC is divided into non-overlapping

16-bit segments, and each segment is declared as a specific coefficient, c_i, $i = 0,1,2,...,8$. Note that this mapping method (from SC to c_i) should be known during decoding, where the inverse operation takes place: decoded coefficients (c_i^*) are mapped back to decoded secret SC*. Then, two sets composed of point pairs need to be generated. The first one, called genuine set G, is found by evaluating $p(u)$ on the template minutiae features (T). Starting with N template minutiae (if we have more than N minutia, we choose the first N sorted according to ascending u values), $u_1, u_2, ..., u_N$, we find $G = \{(u_1, p(u_1)),\ (u_2, p(u_2)),...,\ (u_N, p(u_N))\}$. Note that the template minutiae are selected to be unique, namely, $u_i \neq u_k$, if $i \neq k$, $i = 1,2,...,N$, $k = 1,2,...,N$. The second set, called the chaff set C, determines the security of the system. Assuming we need to add M chaff points, we first generate M unique random points, $c_1, c_2, ..., c_M$ in the field GF(2^{16}), with the constraint that they do not overlap with $u_1, u_2, ..., u_N$, namely $c_j \neq u_i$, $j = 1,2,...,M$, $i = 1,2,...,N$. Then, we generate another set of M random points, $d_1, d_2, ..., d_M$, with the constraint that the pairs (c_j, d_j), $j = 1,2,...,M$ do not fall onto the polynomial $p(u)$. Chaff set C is then $C = \{(c_1, d_1),\ (c_2, d_2),...,\ (c_M, d_M)\}$, where $d_j \neq p(c_j)$, $j = 1,2,...,M$. Union of these two sets, $G \cup C$, is finally passed through a list scrambler which randomizes the list, with the aim of removing any stray information that can be used to separate chaff points from genuine points. This results in vault set, $VS = \{(v_1, w_1),\ (v_2, w_2),...,\ (v_{N+M}, w_{N+M})\}$. Along with VS, the polynomial degree D forms the final vault, V.

3.2 Decoding

Here, a user tries to unlock the vault V using the query minutiae features. Assuming that we have N (note that this number is the same as the number of genuine template minutiae in order to balance the complexity conveyed via the number of required access attempts to reveal the secret) query minutiae (Q), $u_1^*, u_2^*, ..., u_N^*$, the points to be used in polynomial reconstruction are found by comparing u_i^*, $i = 1,2,...,N$, with the abscissa values of the vault V, namely v_l, $l = 1,2,...,(M+N)$: if any u_i^*, $i = 1,2,...,N$ is equal to v_l, $l = 1,2,...,(M+N)$, the corresponding vault point $(v_l,\ w_l)$ is added to the list of points to be used. Assume that this list has K points, where $K \leq N$. Now, for decoding a D-degree polynomial, $(D+1)$ unique projections are necessary. We find all possible combinations of $(D+1)$ points, among the list with size K. Hence, we end up with $C(K, D+1)$ combinations. For each of these combinations, we construct the Lagrange interpolating polynomial. For a specific combination set given as $L = \{(v_1, w_1),\ (v_2, w_2),...,\ (v_{D+1}, w_{D+1})\}$, the corresponding polynomial is

$$p^*(u) = \frac{(u-v_2)(u-v_3)...(u-v_{D+1})}{(v_1-v_2)(v_1-v_3)...(v_1-v_{D+1})}w_1 + \frac{(u-v_1)(u-v_3)...(u-v_{D+1})}{(v_2-v_1)(v_2-v_3)...(v_2-v_{D+1})}w_2 + ...$$
$$... + \frac{(u-v_1)(u-v_2)...(u-v_D)}{(v_{D+1}-v_1)(v_{D+1}-v_2)...(v_{D+1}-v_D)}w_{D+1}$$

This calculation is done in GF(2^{16}) and yields $p^*(u) = c_8^* u^8 + c_7^* u^7 + ... + c_1^* u + c_0^*$. The coefficients are mapped back to the decoded secret SC*. For checking whether there are errors in this secret, we divide the polynomial corresponding to SC* with the CRC primitive polynomial, $g_{CRC}(a) = a^{16} + a^{15} + a^2 + 1$. Due to the definition of CRC, if the remainder is not zero, we are certain that there are errors. If the remainder is zero, with very high probability, there are no errors. For the latter case, SC* is segmented into 2 parts: the first 128-bits denote S* while the remaining 16-bits are CRC data. Finally, the system outputs S*. If the query minutiae list (Q) overlaps with template minutiae list (T) in at least $(D+1)$ points, for some combinations, the correct secret will be decoded, namely, S* = S will be obtained. This denotes the desired outcome when query and template fingerprints are from the same finger. Note that CRC is an error detection method, and it does not leak information that can be utilized by an imposter attacker (Bob). He cannot learn which one of the polynomial projections is wrong; hence he cannot separate genuine points from chaff points.

4 Experimental Results

We used the IBM-GTDB [10] fingerprint database (100 mated image pairs with 500 dpi resolution and approximately of size 300x400) for obtaining the results presented in this section. The minutiae coordinates are linearly mapped to 8-bit range (e.g., the values [0, 255]) for both row and column dimensions before using them in locking/unlocking the vaults. In this database a fingerprint expert has manually marked the minutiae in every image. Further, the expert also established the correspondence between minutiae in mating fingerprint images (two fingerprint images per finger). Using this database has many advantages since (i) the adverse effects of using an automatic (and possibly imperfect) minutiae extractor are eliminated, and (ii) minutiae correspondence has been established. Note that we specifically chose to use such a database because it allows us to establish the upper bound on the performance of the fuzzy fingerprint vault. In our fuzzy vault implementation, the pre-alignment of template and query minutiae sets is based on the correspondence marked by the expert. The translation and rotation parameters that minimize the location error (in the least squares sense) between the corresponding minutiae are found and the query minutiae sets are aligned with template minutiae sets. We randomly generated a 128-bit secret S, and after appending the 16 CRC bits, the resulting 144-bits are converted to the polynomial $p(u)$ as $p(u) = 60467u^8 + 63094u^7 + ... + 52482u^1 + 11995$. As explained in Section 3.1, 144-bit data is divided into 16-bit chunks, and each one of these chunks determines one of the 9 coefficients of $p(u)$. One hundred pairs of fingerprint images (of 100 users) from IBM-GTDB database are used for locking and unlocking the vaults with the following parameters: number of template and query

minutiae $N = 18$ (selected so that the number of candidate points will be enough for reconstruction), number of chaff points $M = 200$ (the effect of this number on the security of the method against attackers is given below), and the block size used for quantization of minutiae *x* and *y* coordinates is 7 pixels (determined experimentally; a larger value decreases the capacity of the vault, whereas a smaller value does not eliminate the minutiae variability). During decoding, 18 query minutiae selected $K = 12$ candidate points, on average. By evaluating the 9-element combinations of these candidate points, 79 of the 100 query fingerprints were able to successfully unlock the vault (that was locked by its mated fingerprint) and recover the secret S. The remaining 21 query fingerprints selected fewer than required number of genuine points (i.e., less than 9) during decoding, hence they were unable to unlock the vault. Hence, the False Reject Rate (FRR) of the proposed system is 0.21, for the cited system parameters (so, the Genuine Accept Rate is 0.79, i.e., 79%). The average number of point combinations required for a genuine user to unlock the vault was 201, which corresponds to 52 seconds of computation for a system with a 3.4 GHz processor. During decoding, many candidate secrets (201 on average) need to be extracted and evaluated, resulting in high time complexity. Further, the system is implemented in Matlab, contributing to high computational times. It is observed that, during decoding, CRC performed with 100% accuracy: it signaled an error *if and only if* there is an error in the decoded polynomial.

For evaluating the corresponding False Accept Rate (FAR), we tried to unlock the vaults with fingerprint templates that *were not* the mates of the templates that locked the vaults. Hence, we have 99 imposter unlocking attempts for a distinct finger, and totally 9900 (99x100) attempts for 100 users. Note that the unlocking templates are first aligned with the locking templates (to avoid giving an unfair disadvantage to imposter attempts), using the centers of mass of minutiae coordinates: the minutiae of unlocking template are translated in *x* and *y* dimensions till their center of mass coincides with the center of mass of locking templates. None of the 9900 attempts could unlock the vault. Hence, for this small database, experimental FAR is 0%.

We can also quantify the security of the system mathematically. Assume that we have an attacker who does not use real fingerprint data to unlock the vault; instead he tries to separate genuine points from chaff points in the vault using brute-force. The vault has 218 points (18 of them are genuine, remaining 200 are chaff); hence there are a total of $C(218,9) \approx 2.6 \bullet 10^{15}$ combinations with 9 elements. Only $C(18,9) = 48620$ of these combinations will reveal the secret (unlock the vault). So, it will take an average of $5.3 \bullet 10^{10}$ (=C(218,9)/C(18,9)) evaluations for an attacker to crack the vault; this corresponds to a computational time of 439 years for the previously used system with the 3.4 GHz processor.

5 Conclusions

After exploring the characteristics of a new cryptographic construct called fuzzy vault, we presented the results of its actual implementation using fingerprint minutiae data, without resorting to simulating the error-correction step. The vault performs as expected (namely, the genuine users can unlock the vault successfully, and the com-

plexity of attacks that can be launched by imposter users is high). It is shown that 128-bit AES keys can be feasibly secured using the proposed architecture. The limitations of our approach include high time complexity (due to the need for evaluating multiple point combinations during decoding). Currently, we are working on architectures for achieving automatic alignment within the fuzzy fingerprint vault.

References

1. W. Stallings, *Cryptography and Network Security: Principles and Practices*, 3. Ed., Prentice Hall, 2003.
2. NIST, Advanced Encryption Standard (AES), 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
3. A. Jain, R. Bolle, and S. Pankanti, Eds., *Biometrics: Personal Identification in Networked Society*, Kluwer, 1999.
4. A. Juels and M. Sudan, "A Fuzzy Vault Scheme", *Proc. IEEE Int'l. Symp. Inf. Theory*, A. Lapidoth and E. Teletar, Eds., pp. 408, 2002.
5. J.-P. Linnartz and P. Tuyls, "New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates", *Proc. 4th Int'l Conf. Audio- and Video-based Biometric Person Authentication*, pp. 393-402, 2003.
6. A. Juels and M. Wattenberg, "A Fuzzy Commitment Scheme", In G. Tsudik, Ed., *Sixth ACM Conf. Computer and Comm. Security*, pp. 28-36, 1999.
7. S. Lin, *An Introduction to Error-Correcting Codes*, Prentice-Hall, 1970.
8. T. C. Clancy, N. Kiyavash, and D. J. Lin, "Secure Smartcard-Based Fingerprint Authentication", *Proc. ACM SIGMM 2003 Multim., Biom. Met. & App.*, pp. 45-52, 2003.
9. U. Uludag and A.K. Jain, "Fuzzy Fingerprint Vault", *Proc. Workshop: Biometrics: Challenges Arising from Theory to Practice*, pp. 13-16, 2004.
10. U. Uludag, S. Pankanti, S. Prabhakar and A. K. Jain, "Biometric Cryptosystems: Issues and Challenges", *Proc. IEEE*, vol. 92, no. 6, pp. 948-960, 2004.
11. A. K. Jain, L. Hong, S. Pankanti, and R. Bolle, "An Identity Authentication System Using Fingerprints", *Proc. IEEE*, vol. 85, no. 9, pp. 1365-1388, 1997.
12. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2. Ed., Cambridge University Press, 1992.