A Practical Implementation of Fuzzy Fingerprint Vault

Sungju Lee[1], Yongwha Chung[1], Daesung Moon[2], Sung Bum Pan[3]and Chang-Ho Seo[4]
[1] Department of Computer and Information Science, Korea University
Seochang 208, Chochiwon, Chungnam, 339-700, Korea
[e-mail : {peacfeel, ychungy}@korea.ac.kr]
[2] ETRI Human Identification Research Team, ETRI
161 Gajeong-dong, Yuseong-gu, Daejeon 305-700, Korea
[e-mail : daesung@etri.re.kr]
[3] Department of Control, Instrumentation and Robot Engineering, Chosun University
375, Seosuk-dong, Dong-gu, Gwangju, 501-759, Korea
[e-mail : sbpan@chosun.ac.kr]
[4] Department of Applied Mathematics, Kongju National University
182, Singwan-dong, Gongju-si, 314-701, Korea
[e-mail : chseo@kongju.ac.kr]
*Corresponding authors: Yongwha Chung and Sung Bum Pan

Received March 31, 2011; revised August 4, 2011; accepted August 15, 2011; published October 31, 2011

Abstract

Recently, a cryptographic construct, called fuzzy vault, has been proposed for crypto-biometric systems, and some implementations for fingerprint have been reported to protect the stored fingerprint template by hiding the fingerprint features. In this paper, we implement the fuzzy fingerprint vault, combining fingerprint verification and fuzzy vault scheme to protect fingerprint templates. To implement the fuzzy fingerprint vault as a complete system, we have to consider several practical issues such as automatic fingerprint alignment, verification accuracy, execution time, error correcting code, etc. In addition, to protect the fuzzy fingerprint vault from the correlation attack, we propose an approach to insert chaffs in a structured way such that distinguishing the fingerprint minutiae and the chaff points obtained from two applications is computationally hard. Based on the experimental results, we confirm that the proposed approach provides higher security than inserting chaffs randomly without a significant degradation of the verification accuracy, and our implementation can be used for real applications.

Keywords: Information security, fingerprint recognition, internet banking/home networking, fuzzy fingerprint vault, correlation attack

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This work was supported by the Korea Science and Engineering Foundation(KOSEF) grant funded by the Korea government(MEST) (No. 2009-0086 148).

DOI: 10.3837/tiis.2011.10.006

1. Introduction

Many of the limitations of password-based key release can be eliminated by incorporating biometric data. It is inherently more reliable than password-based key release 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 feature of authentication. Thus, biometrics-based solution is a potential candidate to replace password-based solution, either for providing complete authentication mechanism or for securing the traditional cryptographic keys. In this paper, the fingerprint has been chosen as the biometrics for user authentication. It is more mature in terms of the algorithm availability and feasibility [1].

However, the fact that fingerprint **templates** are stored in a database and/or a something introduces a number of security risks, and several threats can be identified [2][3][4]: if the fingerprint template storing the user's features (*i.e.*, minutiae) is compromised, the user may quickly run out of his fingers to be used for user authentication and cannot re-enroll forever.

To solve this problem, some researches have reported in order to protect the fingerprint template [5]. For examples, some results using idea of the fuzzy vault scheme [6] have been reported [7][8][9][10][11][12][13]. Juels and Sudan [6] proposed a scheme called **fuzzy vault**. In LOCK function of the fuzzy vault scheme, the **secret** k (such as private key) is encoded as the coefficients of a Galois field **polynomial** $f(x)$. A user's biometric minutiae (set A) are encoded as pairs $(a_i, f(a_i))$, where a_i is a minutia and $f(a_i)$ is a mapping value from the minutia to the polynomial. Additionally, to hide these "**real**" minutiae, numerous "**chaff**" minutiae are encoded, in which the value of $f(a_i)$ is random. During UNLOCK function, new biometric input minutiae (set B) are calculated, and the minutiae a_i closest to the b_i are chosen. The $f(a_i)$ corresponding to these minutiae are used to estimate the polynomial, using a Reed-Solomon error correcting code framework. If enough legitimate (*i.e.*, real) minutiae are taken, the correct polynomial will be obtained and the correct secret k can be decrypted.

Some results using the fuzzy fingerprint vault have been reported [7][8][9][10][11][12][13]. However, these approaches are difficult to apply the realistic for the industrial environment because they do not consider every possible issues such as automatic fingerprint alignment, verification accuracy, execution time, RS decode, etc. For example, Clancy *et al.* [7] proposed a fuzzy fingerprint vault scheme based on the location of minutiae in a fingerprint. However, the False Reject Rate (FRR) of their system was 20~30%. Uludag *et al.* [8] introduced a modification to the fuzzy vault scheme, which uses a simple Cyclic Redundancy Check (CRC) for error-correction. However, they assumed that fingerprints were pre-aligned. Nandakumar *et al.* [9] also proposed the fuzzy fingerprint vault based on the helper data for the auto-alignment. However, this solution also has some limitations such as a slow decoding time or a possibility of failure to find the core point in some images. Also, [14] reported that transform-based cancelable templates may not be secure against the dictionary attack. Therefore, to implement the fuzzy fingerprint vault practically, we have to consider several issues such as automatic fingerprint alignment, verification accuracy, execution time, error correcting code, etc.

In addition, fuzzy fingerprint vault is shown recently to be susceptible to a ***correlation attack*** that correlates the two vaults created using the same finger in order to reveal the fingerprint minutiae hidden in the vault [15][16]. In fact, this type of correlation attack is very

effective for cryptographic schemes which are based on the principle of "chaffing and winnowing" [17]. That is, the security of this scheme relies on a presence of randomly generated chaff that disguises the actual secret data (*i.e.*, fingerprint minutiae) but does not cover the entire universe generated from any fingerprint. Then, the intersection of such vault information gathered from multiple systems (*i.e.*, enrolled for different applications) would likely reveal the secret data [18]. Therefore, we need to cover the entire universe with chaffs in order to make multiple systems look similar.

In this paper, we propose an approach to protect fingerprint templates stored in a database by using the idea of the fuzzy vault [6]. Fuzzy vault is a cryptographic construct and has the characteristics that make it suitable for applications combining biometric authentication and cryptography. Additionally, to avoid the correlation attack [15][16] against the fuzzy vault (*i.e.*, obtains the real minutiae from multiple enrolled vaults), we apply an approach to insert chaffs in a structured way such that distinguishing the fingerprint minutiae and the chaff points obtained from two applications is computationally hard.

The rest of the paper is structured as follows. Section II explains the overview of the fuzzy fingerprint vault and correlation attack. Section III describes the proposed fingerprint template protection approach. The experimental results and conclusions are made in Section IV and V, respectively.

2. Background

2.1 Fingerprint Verification

A typical fingerprint verification system has two phases: *enrollment* and *verification* [1]. In the off-line enrollment phase, an enrolled fingerprint image for each user is preprocessed, and the minutiae are extracted and stored in a server. In the on-line verification phase, the input minutiae are compared to the stored template, and the result of the comparison is returned.

In general, there are three steps involved in the verification phase: Image Pre-Processing, Minutiae Extraction, and Minutiae Matching. Image Pre-Processing refers to the refinement of the fingerprint image against the image distortion obtained from a fingerprint sensor. Minutiae Extraction refers to the extraction of features in the fingerprint image. After this step, some of the minutiae are detected and stored into a pattern file, which includes the position, orientation, and type (ridge ending or bifurcation) of the minutiae. Based on the minutiae, the input fingerprint is compared with the enrolled database in the Minutiae Matching step.

Note that Image Pre-Processing and Minutiae Extraction steps require a lot of integer computations, and the computational workload of both steps occupies 96% of the total workload of the fingerprint verification. When the fingerprint verification phase is implemented into client-server environments, it is reasonable to assign the time-consuming steps to the server, rather than to the resource-constrained client. This kind of task assignment can also solve the *vendor interoperability* problem [19]. That is, the fingerprint minutiae extracted by one vendor may not match well with the stored fingerprint template extracted by another vendor. To solve this vendor interoperability, the fingerprint image itself needs to be transmitted from the handheld device to the server. Also, in order to reduce the size of the transmission, the Wavelet Scalar Quantization (WSQ) fingerprint compression standard can be applied [20]. Finally, there are global feature (*i.e.*, orientation model) based feature extraction methods [21]. Since every fuzzy fingerprint vaults are based on local features (*i.e.*, minutiae), however, we also assume minutiae-based fuzzy fingerprint vault in this paper. The details of global feature based feature extraction methods can be found in [21].

In typical fingerprint verification systems, the fingerprint features are often stored in a central database as an enrolled template. With the central storage of the fingerprint feature, there are open issues of misuse of the fingerprint feature such as the "Big Brother" problem. To solve these open issues [2][3][4][5], new cryptographic constructs combining biometric authentication and cryptography such as fuzzy vault [6] need to be developed.

2.2 Fuzzy Vault Scheme

In the fuzzy vault scheme [6], Alice can place a secret value S in a vault and lock it using an unordered locking set $\boldsymbol{L}$. Bob, using an unordered unlocking set $\boldsymbol{U}$, can unlock the vault only if $\boldsymbol{U}$ overlaps with $\boldsymbol{L}$ to a great extent.

The procedure for constructing the fuzzy vault is as follows: Secret value S is first encoded as the coefficients of some degree k polynomial in x over a finite field $GF(p)$. This polynomial $p(x)$ is now the secret to protect. The locking set $\boldsymbol{L}$ is a set of n values $l_i \in GF(p)$ making up the fuzzy encryption key, where $n>k$. The locked vault contains all the pairs $(l_i, p(l_i))$ and some large number of chaff points (α_j, β_j), where $p(\alpha_j) \neq \beta_j$. After adding the chaff points, the total number of items in the vault is r.

In order to crack this system, an attacker must be able to separate the chaff points from the legitimate points in the vault. The difficulty of this operation is a function of the number of chaff points, among other things. A legitimate user should be able to unlock the vault if they can narrow the search space. In general, to successfully interpolate the polynomial, they have an unlocking set $\boldsymbol{U}$ of m elements such that $\boldsymbol{L} \cap \boldsymbol{U}$ contains at least $k + 1$ elements. The details of the fuzzy vault can be found in [6].

Using the error-correction coding, it is assumed that the legitimate user can reconstruct p (and hence k). The security of the scheme is based on the infeasibility of the polynomial reconstruction problem (*i.e.*, if Bob does not know many points that lie on p, he cannot feasibly find the parameters of p, hence he cannot access k). Since this fuzzy vault can work with unordered sets (common in biometric templates, including fingerprint minutiae data), it is a promising candidate for crypto-biometric systems.

Since the introduction of the fuzzy vault scheme, some implementations results for fingerprint have been reported [7][8][9][10][11][12][13]. Clancy *et al.* [7] proposed a fuzzy fingerprint vault scheme based on the location of minutiae in a fingerprint. However, the False Reject Rate (FRR) of their system was 20~30%. Uludge *et al.* [8] introduced a modification to the fuzzy vault scheme, which uses a simple Cyclic Redundancy Check (CRC) for error-correction, instead of the Reed-Solomon polynomial decoding. Also, the FRR of their system was 21%, and execution time was 52 seconds with a 3.4 GHz processor. An additional problem with these implementations [7][8] is that they assumed fingerprints were pre-aligned. This is not a realistic assumption for fingerprint-based authentication systems, and limits the applicability of their implementations.

Nandakumar *et al.* [9] also proposed the fuzzy fingerprint vault based on the helper data for the auto-alignment. They got a significant improvement in the Genuine Accept Rate (GAR) [9]. However, they have some limitations such as a slow decoding time (*e.g.*, the median and mean decoding time was 3 and 8 seconds, respectively) or a possibility of failure to find the core point in some images. Furthermore, the helper data may leak some information of the fingerprint ridges because the helper data is generated by using the ridge information.

Yang and Verbauwhede [10] aligned automatically two fingerprint templates in the fuzzy vault domain using a concept of reference minutia, which could be generated with the distance and the orientation of two nearest-neighbor minutiae. That is, their system is based on the

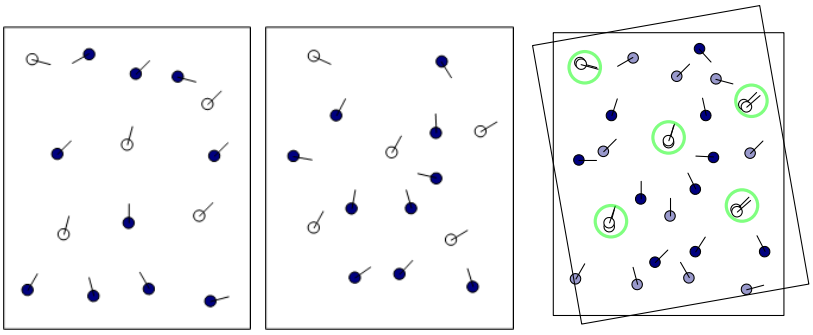
impractical assumption that two reference minutia can always be extracted accurately from both the input and the enrolled fingerprints. They reported a FRR of 17% and FAR of 0% as the verification performance.

Finally, Chung and Moon [11][12][13] proposed an automatic alignment approach by using the geometric hashing technique which has been used for model-based object recognition applications. Note that, the accuracy problem related with rotation transform which may be critical in bio-cryptographic solutions was analyzed in [22]. Also, the difficulty and importance of the alignment problem in fuzzy fingerprint vault was explained in [23]. By using the relative information between minutiae, the geometric hashing technique can solve the geometrical transformation problem like shift and rotation. A similar idea based on composite features was proposed and the details of this solution can be found in [23].

To implement the fuzzy fingerprint vault practically, we have to consider several issues such as automatic fingerprint alignment, verification accuracy, execution time, error correcting code, etc.

2.3 Correlation Attack

Recently, Scheirer *et al*. [15] suggested three critical attacks against fuzzy vaults scheme: i) attacks via record multiplicity, ii) stolen key-inversion attack, and iii) blended substitution attack. Among these attacks, the attack via record multiplicity, called correlation attack [16] as shown in **Fig. 1**, is most important because this type of correlation attack is very effective for cryptographic schemes which are based on the principle of chaffing and winnowing [17]. **Fig. 1 (c)** shows the correlation attack to two different vaults from the same finger.



(a) vault A **(b)** vault B **(c)** result of correlation

Fig. 1. Illustrating the correlation attack to fuzzy fingerprint vault. The correlation attack is (c) with two enrolled fingerprint vaults (a) and (b). White points show fingerprint minutiae, whereas black points show redundant chaff points added randomly to hide the fingerprint minutiae.

In order to show this correlation attack, A. Kholmatov *et al*. [16] reported an implementation of correlation attack to fuzzy fingerprint vaults. For that, they matched each vault pair using exhaustive matching, seeking the best alignment between two vaults over all of their relative rotations and translations. The best alignment is decided as the one which maximizes the number of matching points between the two vaults. After the alignment, two candidate point sets (*CA;CB*) were obtained, one for each of the two vaults, containing a mixture of genuine and chaff points. The vault(s) can be unlocked if i) the number of matching minutiae points M (*i.e.* $L_{CA} \cap L_{CB}$) within a candidate set is sufficient to decode the polynomial of K degree (*i.e.*, $M >= K + 1$) and ii) the total number of matching points N is not too large with respect to M, so that the brute force attack (*i.e.* trying all possible combinations of $K + 1$ out of N points) to reconstruct the polynomial is computationally feasible. During the

experiments, it was observed that 59% of vault pairs were reconstructed by correlation attack, and it took approximately 50 seconds for a non-optimized implementation on a PC with 3GHz CPU.

However, as long as the security of a scheme relies on a presence of randomly generated chaff that disguises the actual secret data but does not cover the entire universe, the intersection of such information gathered from multiple systems would likely reveal the secret [18]. To resist this type of attack, we need to cover the entire universe with chaffs in order to make multiple systems look similar.

3. Proposed Algorithm

In this section, we describe the implementation details of the fuzzy fingerprint and the structured insertion of chaffs in order to protect the correlation attack.

3.1 Implementation of Fuzzy Fingerprint Vault

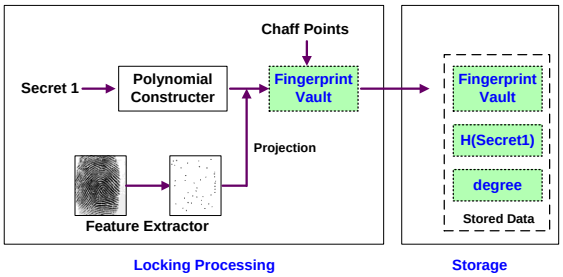
3.1.1 Fuzzy Fingerprint Vault

Minutiae are represented as a point within a 2D fingerprint image. For that reason, the fingerprint is more suitable for applications that combine biometrics and the fuzzy vault scheme than other biometric modalities such as iris and face.

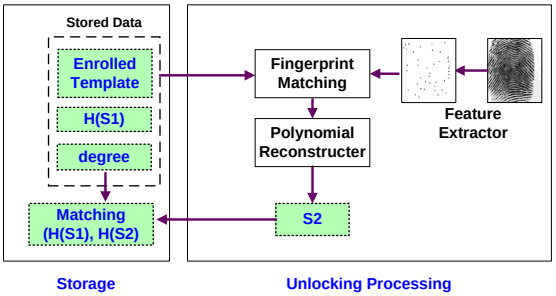
As explained in Section II, a fingerprint minutia represented by $m_i = (x_i, y_i, \theta_i, t_i)$ is composed of four elements: *x-*, *y-coordinates*, *angle*, and *type*. In typical fingerprint verification systems, minutiae are stored in a template file, and input minutiae are compared with the template minutiae after aligning them. In this typical fingerprint verification system, an attacker can reuse the template minutiae once he steals it.

For protecting template minutiae from the attacker, we apply the idea of the fuzzy vault scheme that stores a number of chaff minutiae generated structurally as well as the real minutiae. That is, we consider the template including both real and chaff minutiae and the input minutiae as the locking and the unlocking set, respectively. Then, it is not feasible for the attacker to reuse the fuzzy fingerprint vault directly without separating the real minutiae from the chaff minutiae even if he steals the fingerprint fuzzy vault. Note that, a legitimate user also should not be able to separate the real minutiae without the fingerprint image from the same finger.

Fig. 2 shows our fuzzy fingerprint vault system. As shown in **Fig. 2**, the fuzzy fingerprint vault system is composed of two steps, locking and unlocking. Each step is explained in detail in the following. Note that, a fingerprint minutia represented by $m_i = (x_i, y_i, \theta_i, t_i)$ is composed of four elements: *x-*, *y-coordinates*, *angle*, and *type*.



(a) Locking Process



(b) Unlocking Process

Fig. 2. Illustration of the fuzzy fingerprint vault system

Locking Processing :

① Extract minutiae from the template fingerprint image of a user. These minutiae are called as real minutiae.

$$L=\{(x_i,y_i,\theta_i,t_i)\,|\,i=1,...,n\} \tag{1}$$

where n denote the number of minutiae.

② Generate a degree-k polynomial from a secret S, and compute a hash value κ from a hash function $hash(S)$

$$p(x)=a_0+a_1x+\cdots+a_kx_k \tag{2}$$

$$S=(a_0\,\|\,a_1\,\|\cdots\|\,a_k) \tag{3}$$

$$a_i\in \mathrm{GF}(p^2) \tag{4}$$

$$\kappa=hash(S) \tag{5}$$

③ Compute the polynomial projections $p(x)$ after converting all elements of L to an element of GF(p^2), and define this result as Set R_L. For example, if an element of GF(p^2) is represented as $AX+B$ (A, $B\in$GF(p^2)), we can replace x and y coordinates of the minutia to A and B, respectively.

$$R_L=\{(r_i,v_i)\,|\,i=1,...,n\},\quad r_i=(x_i,y_i,\theta_i,t_i) \tag{6}$$

$$v_i=p(X_i),\quad X_i=x_iX+y_i\in \mathrm{GF}(p^2),\quad i=1,...,n \tag{7}$$

④ Structurally generate chaff minutiae that do not lie on $p(x)$ to protect real minutiae.

$$C=\{(c_i,v_i)\,|\,i=n+1,...,r\},\quad c_i=(x_i,y_i,\theta_i,t_i) \tag{8}$$

$$v_i=p(X_i)+\alpha_i, X_i=x_iX+y_i\in \mathrm{GF}(p^2),\quad i=n+1,...,r \tag{9}$$

where α_I is a non-zero element over finite fields of the form GF(p^2).

⑤ Structurally generate Set R that is integrated with R_L and C.

$$R=\{(r_i,v_i)\,|\,i=1,...,r\},\quad r_i=(x_i,y_i,\theta_i,t_i) \tag{10}$$

⑥ Finally, the vault is constituted by the real and chaff minutiae, the degree k of the polynomial and the secret. The secret should be stored in a hashed form.

$$V=\{(r_i,v_i),\kappa,k\,|\,i=1,...,r\} \tag{11}$$

Unlocking Processing :

Unlocking processing is the step that reconstructs the polynomial from minutiae of the input fingerprint image.

① Extract minutiae from the input fingerprint image. We called set U as unlocking set.

$$U = \{(x'_i, y'_i, \theta'_i, t'_i) \mid i = 1,..., m\} \tag{12}$$

where m denotes the number of minutiae.

② Execute fingerprint matching with Set U and r_i of Set V stored in the Locking processing. The matching results are stored in Set M with t matched minutiae and corresponding v_i in Set V.

$$M = \{(m_i, v_i) \mid i = 1,..., t\}, m_i = (x_i, y_i, \theta_i, t_i) \tag{13}$$

where $M \in R, t \le r$.

③ If k and M are used as input values for the $\mathrm{RS}_{\mathrm{DECODE}}$, the degree-$k$ polynomial $p'(x)$ will be returned. Then, κ' is computed by (16).

$$p'(x) = \mathrm{RS}_{\mathrm{DECODE}}(k, M) \tag{14}$$

$$p'(x) = a'_0 + a'_1 x + \cdots + a'_k x_k \tag{15}$$

$$\kappa' = hash(a'_0 \| a'_1 \| \cdots \| a'_k) \tag{16}$$

④ If κ' and κ are exactly same, the user is accepted. Otherwise, he is rejected.

$$\mathrm{Decision} = \begin{cases} \mathrm{Accept}, & if \quad \kappa' = \kappa \\ \mathrm{Reject}, & otherwise \end{cases} \tag{17}$$

If set M contains k+1 real minutiae, the fuzzy fingerprint vault can reconstruct the same polynomial used in the locking process. However, the chaff minutiae should be removed by using an error correct code such as Reed Solomon (RS) decode because set M may contain some chaff minutiae as well as the real minutiae (even if the user is genuine).

The RS decoding algorithm can remove these chaff minutiae and reconstruct a polynomial using a set of over-sampled minutiae that contains a reasonable number of errors [8]. To reconstruct the k degree polynomial, the RS decode requires at least (k+t)/2 real minutiae (*i.e.*, t is the number of matched minutiae). Also, the performance of the RS code is very important for real applications.

3.1.2 Automatic Alignment

In the previous result [11], the geometric hashing was used to provide a scalable performance for one-to-many matching of fingerprints on large-scale databases. Chung and Moon [11][12][13] proposed the approach to solve the auto-alignment problem in the fuzzy fingerprint vault using the idea of the *geometric hashing* [24]. This is reasonable because the idea of the geometric hashing can also be used for pre-computing the possible alignments to save time for alignment in the one-to-one fuzzy fingerprint vault problem.

In typical fingerprint verification systems, minutiae are stored in the template file, and input minutiae are compared with the template minutiae after aligning them. Similarly, the geometric hashing consists of two procedures – preprocessing (or enrollment) and recognition (or identification).

The *preprocessing* procedure is executed off-line and only once. In this procedure, the model features are encoded and are stored in a hash table. The information is stored in a highly redundant multiple-viewpoint way. Assume each model in the database has n features. For each ordered pair of features in the model chosen as a basis, the coordinates of all other features in the model are computed in the orthogonal coordinate frame defined by the basis pair. Then, (model, basis) pairs are entered into the hash table bins by applying a given hash function f to the transformed coordinates.

In the *recognition* procedure, a scene consisting of S features is given as input. An arbitrary ordered pair of features in the scene is chosen. Taking this pair as a basis, the coordinates of

the remaining features are computed. Using the hash function on the transformed coordinates, a bin in the hash table (constructed in the preprocessing procedure) is accessed. For every recorded (model, basis) pair in the bin, a vote is collected for that pair. The pair winning the maximum number of votes is taken as a matching candidate. The execution of the recognition procedure corresponding to one basis pair is termed as a *probe*. If no (model, basis) pair scores high enough, another basis from the scene is chosen and a different probe is performed.

When we apply this geometric hashing to the fuzzy vault, we should perform 1:1 comparisons. Thus, we use the notion of *verification*, instead of identification. After the enrollment procedure, the verification procedure separates the chaff minutiae *C* from the real minutiae *G* in the *enrollment minutiae table*. That is, the minutiae information (unlocking set *B*) of a verification user is computed and a table, called *verification minutiae table*, is generated according to the geometric characteristic of the minutiae. Then, the verification minutiae table is compared with the enrollment minutiae table, and the subset of real minutiae is finally selected. Note that, the verification minutiae table is generated in the same way as the enrollment procedure. **Fig. 3** shows an illustration of the processing of the hash table generation and the auto-alignment processing with enrollment and input hash table.

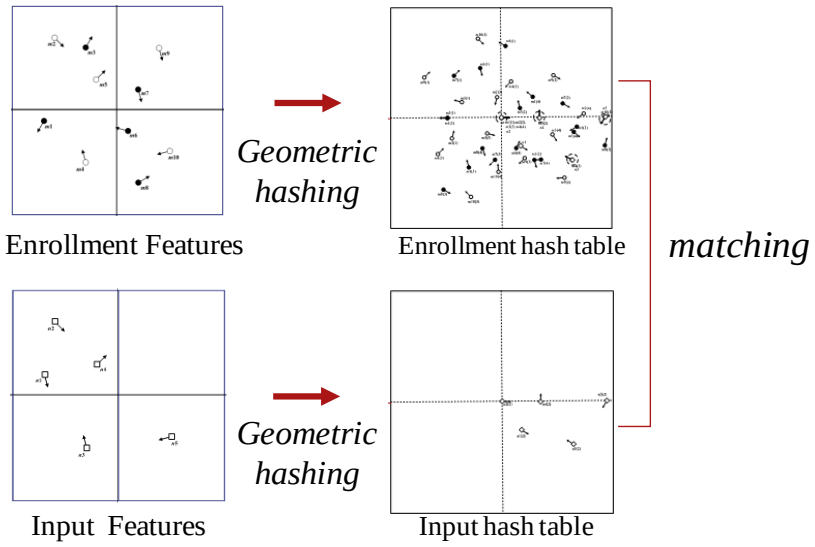


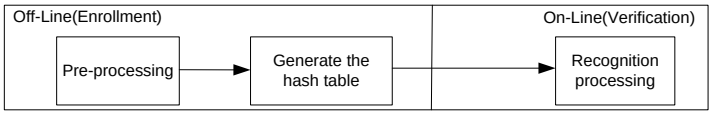
Fig. 3. An Illustration of the Auto-Alignment Processing with Enrollment and Input Hash Table [11].

Although the FFV using the geometric hashing technique can solve the alignment problem, it requires large memory space at the enrollment phase. Also, the enrolment hash table increases with the number of the chaff minutiae inserted as well as the number of the real minutiae. Thus, inserting more chaff minutiae for higher security requires a larger hash table.

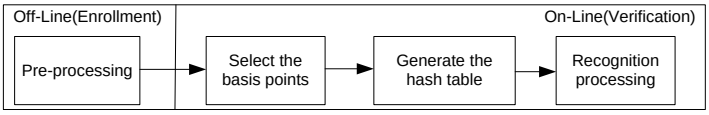
Therefore, to reduce the size of hash table, we implemented the automatic alignment in the fuzzy fingerprint vault by using memory-efficient geometric hashing technique [13][25]. That is, in order to reduce the static memory requirement, the hash table is generated "on-the-fly" at the verification phase (on-line), instead of the enrollment phase (off-line). With this solution, we can reduce the fingerprint template size up to the minimum.

Fig. 4 compares the auto-alignment of fuzzy fingerprint vault based on the geometric hashing technique [11][12] with memory-efficient geometric hashing technique. Since the hash table is generated on-the-fly, the execution speed may be degraded. With the careful selection of the basis set, however, this solution can avoid the significant increase in the

execution time as well as reduce the dynamic memory requirement [13][25]. Additional details can be found in [11][12][13][24][25][26].



(a) Geometric hashing technique [11][12]



(b) Memory-efficient geometric hashing technique [13][25]

Fig. 4. Comparison of Geometric Hashing Technique.

3.1.3 Reed-Solomon Decoder

To completely reconstruct the secret locked within the fuzzy vault, the points in the unlocking set (*i.e.*, contain both real and chaff points) must be used to interpolate a polynomial. Polynomial interpolation method is to use a standard error correcting code such as hamming or Reed-Solomon. We implemented RS code suggested by Jules and Sudan [6]. Note that, although all the previous results assumed RS code, they have not implemented the RS code in fuzzy fingerprint vault [7][10][11][12][13] or used CRC to verify their solutions [8][9]. While RS codes are traditionally used to correct errors in messages transmitted over noisy channels, they essentially a generalization of the polynomial reconstruction problem. From a generalized point of view, the encoding algorithm for the RS code is a repeated sampling of a polynomial defined over a finite field, and the decoding algorithm is a reconstruction of this polynomial using a set of over-sampled points that contains a reasonable number of errors.

While the encoding procedure is straightforward, there are a few well-known approaches for the decoding procedure. The simplest mechanism for polynomial reconstruction is an exhaustive search. That is, for every possible combination of k points out of the set of r points ($r>k$), we interpolate a polynomial with degree k-1 and then select the most probable polynomial using a majority vote. Though this approach is easy to implement, its execution requires more computational power to deal with all the possible cases. Therefore, more elegant decoding algorithms such as the Berlekamp-Massey algorithm [27] and the Guruswami-Sudan algorithm [28] need to be considered.

More recently, Gao [29] proposed a new RS decoding algorithm based on the Extended Euclidean Algorithm. From our viewpoint, the Gao's algorithm has several desirable features as follows:

- It directly reconstructs the original polynomial without finding syndromes, error locations, and error magnitudes as the Berlekamp-Massey algorithm does. Hence, it is more intuitive in the sense that the target polynomial is visible in the whole decoding process.
- It is composed of efficient operations such as interpolation of a polynomial without error, partial GCD computation, and a polynomial division. Hence, it is easier to implement than the Guruswami-Sudan algorithm, with the reasonable performance guaranteed.

Now, we consider the underlying field for the RS code. Though any finite field can be used, finite fields of the form GF(p^2) seem to be ideal for our purpose as is suggested in [7]. This is because the information being encoded by the RS code is pixel coordinates (x_i, y_i) of fingerprint minutiae. To be more precise, we use $p = 2^{16}-17 = 65{,}519$ for the characteristic.

Thus, we can support almost 16-bit resolutions for x and y coordinates for a fingerprint minutia, fully utilizing the integer operations over a 32-bit architecture.

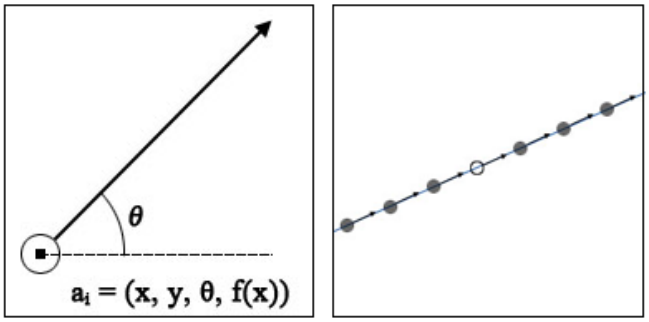
3.2 Structured Insertion of Chaffs

In the proposed approach, the chaffs are generated by using the user's minutiae direction information. To avoid the correlation attack (*i.e.*, to make multiple systems look similar), we structurally inserts chaffs into a vault, instead of randomly, by using the direction information.

Fig. 5 shows the illustration of the structured insertion. **Fig. 5 (a)** shows that a minutiae consists of the location of the x-y coordinates, direction information θ, and projection $f(x)$. To structurally insert chaffs, **Fig. 5 (b)** shows an example of inserting chaffs using direction θ, and **Fig. 6** shows an example of the result of structured insertion of chaffs. For example, a minutia generates a line using direction θ, then chaffs are generated with having the same direction and inserted along the line.

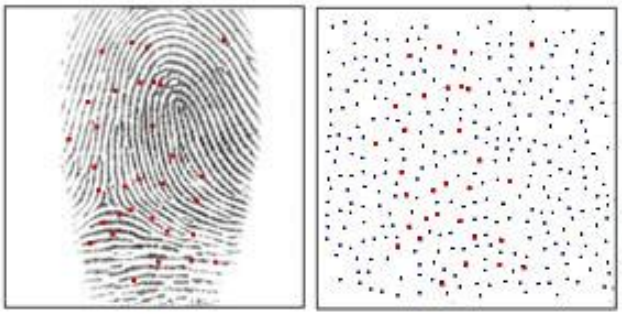
Note that, to guarantee both the fingerprint verification accuracy and the security level simultaneously, we should be careful to determine the distance d_t between each point along the line. Then, Δx and Δy can be calculated by using the minutia's direction information θ and d_t as follows. For each user's minutia, the chaffs are inserted along the line by adding/subtracting Δx and Δy to/from the x-y coordinates of the minutia. However, both the location and the orientation of minutia are not changed.

$$\begin{cases} \Delta \mathrm{x} = \cos\theta \times d_t \\ \Delta y = \sin\theta \times d_t \end{cases} \tag{18}$$



(a) **(b)**

Fig. 5. Chaff insertion by using the minutiae information : **(a)** minutiae information, **(b)** inserted chaffs along the line generated by the direction information (●: chaff minutiae, ○: real minutiae).



(a) **(b)**

Fig. 6. The result of chaffs inserted : **(a)** the real minutiae extracted from a fingerprint image, **(b)** the result of structured insertion of chaffs.

4. Experimental Results

In this section, we describe the performance of both security level and verification accuracy. We used Set A of the FVC2002-DB1 fingerprint database [30]. The average number of minutiae was 30, and we inserted about 200, 300, and 400 chaffs, respectively. Also, since the fingerprint authentication system can be assumed to request a user who inputs a low quality fingerprint image to re-enter fingerprint image, we excluded low quality fingerprint images. Therefore, among the 8 impressions, we used only two impressions of good image quality for each finger 200 images in total. All experiments were performed on a system with a 2.66 GHz processor.

Table 1 and Table 2 show the number of instructions and the execution times of the locking process and the unlocking process, respectively. As shown in **Table 1**, the execution time of the locking process is about 0.12 second on a 2.66 GHz processor. Especially, the locking process can be executed in real-time, regardless of the number of chaff minutiae.

Table 1. Experimental result of locking process.

# of Chaff	200	300	400
Feature Extraction	0.119 s	0.119 s	0.119 s
Creating Polynomial	0.000 s	0.000 s	0.000 s
Adding Chaff	0.001 s	0.001 s	0.001 s
Total	0.120 s	0.120 s	0.120 s

The time required to unlock the vault is about 0.3, 0.46, and 0.7 second in case of adding 200, 300, and 400 chaff minutiae, respectively. As shown in **Table 2**, in spite of adding 400 chaff minutiae in the fuzzy fingerprint vault, the careful implementation of the proposed approach can guarantee the real time execution.

Table 2. Experimental result of unlocking process.

# of Chaff	200	300	400
Feature Extraction	0.119 s	0.119 s	0.119 s
Matching	0.129 s	0.217 s	0.369 s
RS decode	0.001 s	0.001 s	0.001 s
Total	0.300 s	0.469 s	0.706 s

Table 3 shows a comparison of several implementations of the fuzzy fingerprint vault [7][8][9][10][11][12] including the proposed approach. In the first column, manual and automatic alignment are denoted by M and A, respectively. Also, the second column indicates the real minutiae (R), chaff minutiae (C), and helper data (H). It is difficult to compare the experimental results directly because the experimental environments (*e.g.*, fingerprint database, processor capacity, etc) are different with each other. For example, the proposed approach had fast unlocking execution time than other approaches [8][9][11][12], nevertheless this approach performed the geometric hashing. Also, using memory-efficient geometric hashing, we can reduce the template size, and improve the execution time with careful selection of a basis [13][23]. Therefore, the proposed approach can provide the acceptable performance as shown in **Table 3**, and can be applied to real applications.

Table 3. Comparison of the Previous Researches and the Proposed Approach.

	Align	Template Size	ECC	Unlocking Time (sec)	Verification Accuracy	
					FAR	FRR
Clancy [7]	M	R+C	N/A	N/A	N/A	N/A
Uludag [8]	M	R+C	CRC	52	0	21
Nandakumar [9]	A	R+C+H	CRC	3	0.01	5
Yang [10]	A	R+C	N/A	N/A	N/A	17
Chung [11] Moon [12]	A	(R+C)×(R+C-1)	N/A	1.3 [11]	0.16 [12]	5 [12]
Proposed	A	R+C	RS decode	0.7	0.5	6

Additionally, for the purpose of evaluating the security level against the correlation attack, we compared the performance of the structured chaff insertion approach (*i.e.*, the chaffs are generated by using the user's minutiae direction information) with that of random chaff insertion approach (*i.e.*, we implemented it based on [8] to evaluate the security level). In order to determine the security level against the correlation attack (*i.e.*, $security_{CA}$), we first define some notations. $Reconstruct_{CA}$ is defined as the number of evaluations (*i.e.*, C(no. of minutiae + no. of chaffs obtained from the correlation attack, degree+1)/C(no. of minutiae obtained from the correlation attack, degree+1)) required for an attacker to crack the vaults by applying the correlation attack. $Probability_{CA}$ is defined as the probability of obtaining the minutiae enough to reconstruct the polynomial by applying the correlation attack. Then, $security_{CA}$ can be represented by $reconstruct_{CA}/probability_{CA}$.

Table 4 compares the security level and verification accuracy of different chaff insertion approaches. Based on the experimental results, we confirmed that the proposed approach can improve the security level by a factor of 153 against the correlation attack without significant degradation of the verification accuracy.

Table 4. Performance comparison against the correlation attack.

About 300 chaffs	Security		Verification Accuracy	
	Brute-force Attack	Correlation Attack	FAR	FRR
Random Insertion of Chaffs	2.93×10^{7}	1.69×10^{4}	0.2	7.5
Structured Insertion of Chaffs		2.59×10^{6}	0.5	6.0

5. Conclusions

To implement the fuzzy fingerprint vault practically, we have to consider several issues such as automatic fingerprint alignment, verification accuracy, execution time, error correcting code. We have implemented a fuzzy fingerprint vault as a complete system. Additionally, to avoid the correlation attack against the fuzzy vault, we applied an approach to insert chaffs in a structured way such that distinguishing the fingerprint minutiae and the chaff points obtained

from two applications is computationally hard. Based on the experimental results, we confirmed that the proposed approach can be resistant to the correlation attack (by a factor of 153) as well as satisfy the verification accuracy requirement.

References

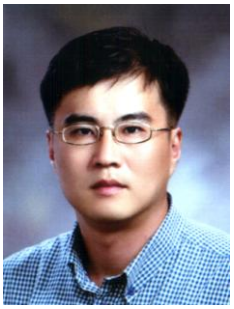
[1] D. Maltoni, et al., Handbook of Fingerprint Recognition, 2003.
[2] R. Bolle, J. Connell, and N. Ratha, "Biometric Perils and Patches," *Pattern Recognition*, vol. 35, pp. 2727-2738, 2002. Article (CrossRef Link)
[3] B. Schneier, "The Uses and Abuses of Biometrics," *Communications of the ACM*, vol. 42, no, 8, pp. 136, 1999. Article (CrossRef Link)
[4] S. Prabhakar, *et al.*, "Biometric Recognition: Security and Privacy Concerns," *IEEE Security and Privacy*, pp. 33-42, 2003. Article (CrossRef Link)
[5] J. Hu, "Mobile Fingerprint Template Protection: Progress and Open Issues," in *Proc. of IEEE ICIEA Conference*, pp. 3-5, 2008. Article (CrossRef Link)
[6] A. Juels and M. Sudan, "A Fuzzy Vault Scheme," in *Proc. of Symp. on Information Theory*, pp. 408, 2002. Article (CrossRef Link)
[7] T. Clancy, *et al.*, "Secure Smartcard-based Fingerprint Authentication," in *Proc. of ACM SIGMM Multim., Biom. Met. & App*., pp. 45-52, 2003. Article (CrossRef Link)
[8] U. Uludag, et al., "Fuzzy Vault for Fingerprints," in *Proc. of Audio- and Video-based Biometric Person Authentication*, pp. 310-319, 2005. Article (CrossRef Link)
[9] K. Nandakumar, *et al.*, "Fingerprint-based Fuzzy Vault: Implementation and Performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744-757, 2007. Article (CrossRef Link)
[10] S. Yang and I. Verbauwhede, "Automatic Secure Fingerprint Verification System Based on Fuzzy Vault Scheme," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 5, pp. 609-612, 2005. Article (CrossRef Link)
[11] Y. Chung, *et al.*, "Automatic Alignment of Fingerprint Features for Fuzzy Fingerprint Vault," *LNCS* 3822, pp. 358-369, 2005. Article (CrossRef Link)
[12] D. Moon, *et al.*, "Fingerprint Template Protection Using Fuzzy Vault," *LNCS* 4707, pp.1141-1151, 2007. Article (CrossRef Link)
[13] D. Moon, *et al.*, "Configurable Fuzzy Fingerprint Vault for Match-on-Card System," *IEICE Electron Express*, vol. 6, no. 14, pp. 993-999, 2009. Article (CrossRef Link)
[14] S. Shin, M. Lee, D. Moon, and K. Moon, "Dictionary Attack on Functional Transform-based Cancelable Fingerprint Templates," *ETRI Journal*, vol. 31, no. 5, pp. 628–630, 2009. Article (CrossRef Link)
[15] W. Scheirer and T. Boult, "Cracking Fuzzy Vaults and Biometric Encryption," in *Proc. of IEEE Biometrics Research Symposium at the National Biometrics Consortium Conference*, 2007. Article (CrossRef Link)
[16] A. Kholmatov and B. Yanikoglu, "Realization of Correlation Attack against the Fuzzy Vault Scheme," in *Proc. of SPIE*, vol. 6819, pp. 68190O.1-68190O.7, 2008. Article (CrossRef Link)
[17] R. Rivest, *et al.*, "Chaffing and Winnowing: Confidentiality without Encryption," http://theory.lcs.mit.edu/~rivest/chaffing.txt, 1998.
[18] D. Socek, D. Culibrk, and V. Bozovic, "Issues and Challenges in Storing Biometric Templates Securely," in *Proc. of CRISIS*, pp. 75-81, 2007. Article (CrossRef Link)
[19] N. Ratha and R. Bolle, "Automatic Fingerprint Recognition Systems," Springer, 2004.
[20] T. Hopper, C. Brislawn, J. Bradley, "WSQ Gray-Scale Fingerprint Image Compression Specification," *Federal Bureau of Investigation*, Document No. IAFIS-IC-0110 (v2), 1993. Article (CrossRef Link)
[21] Y. Wang, J. Hu, and D. Philip, "A Fingerprint Orientation Model Based on 2D Fourier Expansion (FOMFE) and its Application to Singular-point Detection and Fingerprint Indexing," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 4, pp. 573-585, 2007. Article (CrossRef Link)
[22] P. Zhang, J. Hu, C. Li, M. Bennamound, and V. Bhagavatulae, "A Pitfall in Fingerprint Bio-Cryptographic Key Generation," *Computers and Security*, Elsevier, 2011. Article (CrossRef Link)
[23] K. Xi and J. Hu. "Biometric Mobile Template Protection: A Composite Feature Based Fingerprint Fuzzy Vault," in *Proc. of IEEE ICC*, 2009. Article (CrossRef Link)
[24] H. Wolfson and I. Rigoutsos, "Geometric Hashing: an Overview," *IEEE Computational Science and Engineering*, vol. 4, pp. 10-21, Oct.-Dec. 1997.
[25] S. Lee, *et al.*, "Memory-Efficient Fuzzy Fingerprint Vault Based on the Geometric Hashing," in *Proc. of ISA*, pp. 312-315, 2008. Article (CrossRef Link)
[26] S. Chae, S. Lim, S. Bae, Y. Chung, and S. Pan, "Parallel Processing of the Fuzzy Fingerprint Vault based on Geometric Hashing," *KSII Tr. Internet & Info Systems*, vol. 4, no. 6, pp. 1294-1310, 2010. Article (CrossRef Link)
[27] E. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968. (Revised edition, Laguna Hills: Aegean Park Press, 1984.
[28] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic Geometric Codes," in *Proc. of FOCS*, pp. 28-39, 1998.
[29] S. Gao, "A New Algorithm for Decoding Reed-Solomon Codes," *Communications, Information and Network Security*, pp. 55-68, Kluwer Academic Press, 2003.
[30] http://bias.csr.unibo.it/fvc2002/databases.asp.

Sungju Lee received his B.S. and M.S. degrees from Korea University, Korea in 2006 and 2008, respectively. He is currently in Ph.D. program in the Department of Computer and Information Science at The Korea University. His research interests include biometrics, information security, and energy-efficiency of image compression

Yongwha Chung received the BS and MS degrees from Hanyang University, Korea, in 1984 and 1986. He received the PhD degree from the University of Southern California, USA in 1997. He worked for ETRI from 1986 to 2003 as a Team Leader. Currently, he is a Professor in the Department of Computer Information, Korea University. His research interests include biometrics, security, and performance optimization.

Daesung Moon received his M.S. degrees from Busan National University, Korea in 2001. He received his Ph.D. degree from the University of Korea, Korea in 2007. He joined ETRI in 2001 and he has been a senior member of research staff at Human Identification Research Team. His research areas are intelligent video surveillance, biometric, image processing, and security.



Sung Bum Pan received his Ph.D. degrees in Electronics Engineering from Sogang University, Korea, in 1999, respectively. He was a team leader at Biometric Technology Research Team of ETRI from 1999 to 2005. He is now professor at Chosun University. His current research interests are biometrics, security, and VLSI architectures for real-time image processing.

Changho Seo received his B.S., M.S. and Ph.D. degrees from Korea University, Korea in 1990 and 1992 and 1996 in the Department of Mathematics at Korea University, respectively. His research interests include cryptography, information security, and system security.