A Simple Fingerprint Fuzzy Vault for FIDO

Dongil Cho
R&D Center, Tomato System
Nonhyeon-ro 128-gil, Gangnam-Gu, Seoul 06104 - Korea
[e-mail: chodongil@yahoo.co.kr]
*Corresponding author: Dongil Cho

Received April 18, 2015; revised July 23, 2016; revised September 12, 2016; accepted October 15, 2016; published November 30, 2016

Abstract

Fast IDentity Online(FIDO) supports biometric authentications in an online environment without transmitting biometric templates over the network. For a given FIDO client, the "Fuzzy Vault" securely stores biometric templates, houses additional biometric templates, and unlocks private keys via biometrics. The Fuzzy Vault has been extensively researched and some vulnerabilities have been discovered, such as brute force, correlation, and key inversions attacks. In this paper, we propose a simple fingerprint Fuzzy Vault for FIDO clients. By using the FIDO feature, a simple minutiae alignment, and point-to-point matching, our Fuzzy Vault provides a secure algorithm to combat a variety of attacks, such as brute force, correlation, and key inversions. Using a case study, we verified our Fuzzy Vault by using a publicly available fingerprint database. The results of our experiments show that the Genuine Acceptance Rate and the False Acceptance Rate range from 48.89% to 80% and from 0.02% to 0%, respectively. In addition, our Fuzzy Vault, compared to existing similar technologies, needed fewer attempts.

Keywords: Fuzzy Vault Scheme, FIDO, Biometric, Fingerprint, Authentication

http://dx.doi.org/10.3837/tiis.2016.11.026 **ISSN : 1976-727**

1. Introduction

The Fast IDentity Online (FIDO) Alliance is an association established to determine the technical standards for authentication using biometric technology in an online environment [1]. The specifications of FIDO should help support a full range of authentication technologies, including fingerprints and other biometric modalities [2]. FIDO authentication uses public-key cryptography, where a device registers the user to a server via a public key. The device authenticates the user by signing a challenge from the server using the private key it holds. The private keys on the device are unlocked by a local user gesture, such as a biometric.

However, the FIDO standard does not specify methods to store biometric templates and to authenticate users. Authentication on a FIDO client requires fingerprint templates to be stored on the user's device. The encrypted fingerprint template should be decrypted for similarity measures when authenticated and the secure key should be managed. This key can be inputted by the user, which complicates the authentication and is against the fundamental purpose of FIDO. In this regard, a Fuzzy Vault can be considered a suitable technology. The Fuzzy Vault scheme is a cryptographic primitive that can be used to protect a user's fingerprint template and can unlock a secret κ, by using similar keys [3].

There have been numerous studies on Fuzzy Vault since it was proposed in [4]. Our literature review uncovered several susceptibilities of Fuzzy Vaults against various discrete intrusions, including brute force, correlation, and key inversion attacks [3][5][6][7][8].

This prompted us to propose a simple Fuzzy Vault for FIDO authentications. Our simple Fuzzy Vault reduces security issues compared to existing technologies proposed in previous studies and provides a simpler structure, which is interoperable alongside the FIDO client.

2. Related Work

The Fuzzy Vault scheme proposed by Juels and Sudan [4] locks and unlocks the polynomial $\mathcal{P}$ it wants to protect by using two keys with certain similarities. In [4], Alice places a secret κ in a Fuzzy Vault and locks it using a set G of elements in a public universe U. To unlock the vault and retrieve κ, Bob must present a set Q that substantially overlaps with G. Fuzzy Vaults are order invariant, meaning G and Q may be arranged in any order. A Fuzzy Vault protects κ by encoding it within the coefficients of polynomial $\mathcal{P}$. A set of points V is constructed from G and $\mathcal{P}(G)$. In addition, chaff points C are randomly generated and inserted into V. These authors [4] solve the subset matching problem with Reed-Solomon coding. The following rule is used to decode κ: If Q approximately matches G and there is a sufficient number of points in V that lie on $\mathcal{P}$, then applying an error correcting code can reconstruct $\mathcal{P}$ to reveal κ [6].

After this study, others [9][10][11] proposed an algorithm that excludes error correction from the Fuzzy Vault to avoid the complexity of Reed-Solomon codes and adds verification through the Cyclic Redundancy Check (CRC) code.

In [12], genuine minutiae made a one-time transformation by using randomized rotation matrices, vectors, and other relevant information, which are then saved on a smartcard or a security token. During verification, query minutiae are transformed by using saved options and are compared with the vault stored in the database. Nonetheless, for this particular study, there was a noticeable absence of chaff point creations and a method of matching.

The security risks associated with Fuzzy Vault have also been explored. More specifically, brute force, correlation, and key inversion attacks have been reported [3][5][6][7]. In [5], an algorithm that solves correlation attacks from previous Fuzzy Vaults that utilize Reed-Solomon code and additional hashing algorithms was presented. As noted in [3], when a specific absolute pre-alignment was applied, the Fuzzy Vault was free of various security risks.

However, previous studies that have used CRC, Reed-Solomon, or hash code for verification of recovered options are not considered to have these security threats [4][9][10][11][12][13]. These studies were unable to provide solutions that were resistant to brute force attacks because the attackers were still able to verify the authenticity of candidate points [3][8][14][15]. Moreover, the pre-alignment for increasing the similarities between two minutiae (query and genuine) and verifications were not able to perform exact matches; therefore, the additional noise caused an increase in the False Acceptance Rate (FAR). Furthermore, previous approaches were vulnerable to correlation attacks. Finally, the above-mentioned experiments from previous studies were only tested with a small amount of fingerprint minutiae and a small-sized κ [9][10][13][14].

FIDO authentication and real-life applications require the Fuzzy Vault to be able to (1) accept and process more minutiae and (2) protect κ against brute force, correlation, and key inversion attacks. Additionally, the size of κ for encrypting a user private key is supported with 256 bits or more in the FIDO client.

3. Fuzzy Vault Implementation

Our simple Fuzzy Vault has the following features :

- The secret κ has at least 256 bits
- Supports at least 32 minutiae
- Invulnerability against brute force, correlation, and key inversion attacks

In this paper, we provide an overview of the Fuzzy Vault. However, an elaborate explanation about how the Fuzzy Vault interlocks with components other than the internal process of the FIDO client is omitted because the FIDO protocol, amongst other

components, is described in detail within the FIDO specification. The Fuzzy Vault scheme consists of two stages – enrollment and authentication. In this section, we will describe their respective details.

3.1 Enrollment

Fig. 1 illustrates the enrollment flow of our simple Fuzzy Vault.

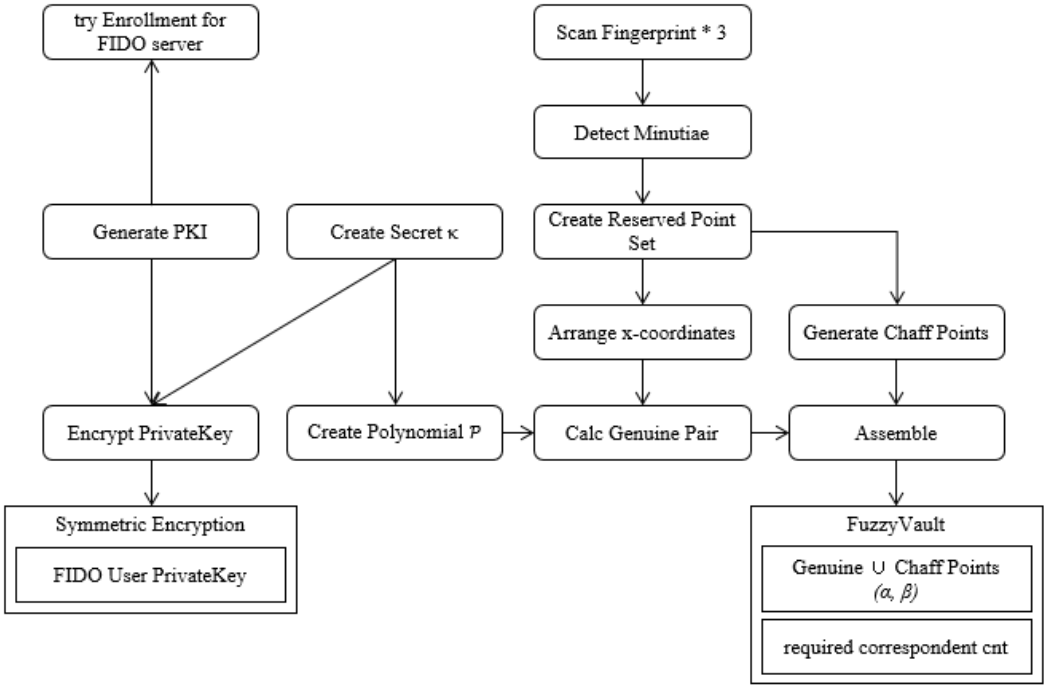

Fig. 1. Enrollment flow

Enrollment consists of the following procedures:

- Detection of the minutiae set A, A', and A'' from three different images of a given finger,
- Alignment and transformation of detected minutiae into a reserved point set R,
- Creation of the secret κ and public and private keys,
- Encryption of the private key while adopting κ,
- Construction of polynomial P with κ as its coefficients,
- Selection of a genuine point set G among A, A', and A'',
- Adding the chaff point set C into the Fuzzy Vault by using R, and

• Creating a Fuzzy Vault that consists of G, C, and the minimum number of correspondent points τ

After these steps have been completed, the public key is sent to the FIDO server, following which the key is associated with the user's account. The private key and other pertinent information about the local authentication method (e.g., biometric measurements or templates) never leave the local device [2].

Note that any code for verification or error-correction is not included.

■ Minutiae Detection

In our implementation, we employed the MINDTCT of Biometric Image Software of NIST (NBIS) for the extraction of minutiae. The MINDTCT outputs the result of minutiae extraction as [16] : *{MN : MX, MY : DIR : REL : TYP : FTYP : FN : NX1, NY1; RC1 : ...}* of which we use *MN*, *MX*, *MY*, *DIR*, *REL*, and *TYP*.

For a detected minutia, *MN* represents the integer identifier, *MX* stands for the x-pixel coordinate, *MY* expresses the y-pixel coordinate, and *DIR* serves as the direction.

DIR is a uni-directional starting point that vertically points up with unit 0, which increases and rotates clockwise in increments of 11.25 degrees, completing a full circle.

REL is the reliability measurement assigned to the detected minutia. This measure is computed by looking up the quality level associated with the position of the minutia from the Quality Map of MINDTCT. The quality level is then heuristically combined with simple neighborhood pixel statistics surrounding the minutia point. The result is a floating point value in the range of 0.0 to 1.0, with 0.0 representing the lowest minutia quality and 1.0 representing the highest minutia quality.

TYP is the type of the detected minutia and is a bifurcation or ridge ending. In [8][9][10] and [17], a ridge ending and ridge bifurcation are considered to be the same, and are used for encoding and decoding of the Fuzzy Vault. In fact, both could be represented as *(x, y, θ)* triplets, but they are different features. Thus, to improve the accuracy, a comparison with identical features should be done. However, in order to use these features together, a high computational complexity and an additional mechanism are needed. In this study, we only use a ridge ending to focus on the matching algorithm and the method of chaff point creation.

■ Creation of a Reserved Point Set

We created a chaff point in the XY plane. These points can exist on the actual fingerprint image in order to increase the similarities between G and C inside the Fuzzy Vault. At this step, three different images from a fingerprint are used to prevent overlap query and chaff points. Minutiae A, A', and A'' are extracted from three fingerprint images and are overlapped on one XY plane. Following this, our algorithm processes the data into a

reserved point set *R*, which is used as a primary restriction for creating chaff points.

■ Point-to-Point Matching

Multiple existing studies on Fuzzy Vault implement pre-alignment within the enrollment and authentication process [3][9][10]. Despite such implementations, pre-alignment is impractical in real-world scenarios and can contribute to the rise of an False Rejection Rate (FRR) [6][17]. In conjunction with the extraneous information stored for restoring alignment, correlation matching becomes much more susceptible to correlation attack attempts [3][17].

Our matching algorithm applies point-to-point matching without this unnecessary pre-alignment step and is described in **Algorithm. 1.**

Algorithm. 1. Point-to-Point Matching Algorithm

Public Parameter : d_{max}, θ_{max}
Input : Target Point Set $T = \{t_i\}_{i=1}^{n}$
Query Point Set $Q = \{q_j\}_{j=1}^{l}$
Output : A set *M*, $M \subset T$

$M, C \leftarrow \emptyset;$
$for\ i = 1\ to\ n$
$for\ j = 1\ to\ l$
$g = GAP(t_i, q_j);$
$if\ g < (d_{max}, \theta_{max})$
$Q' \leftarrow ALIGN(q_j, g, Q);$
$M \leftarrow FILTER(T, Q');$
$if(sizeOf(M) \geq \tau)\ output\ M;$
$end\ if;$
$end\ for;$
$end\ for;$

The function *GAP* computes distance d_{tq} and angle θ_{tq} between a point q_j and a point t_i. Both q_j and t_i consist of position (*x*, *y*) in the XY plane and direction (θ) of minutia. The function *GAP* can be expressed as

$$GAP(t_i, q_j) = \left(\sqrt{(x_q - x_t)^2 + (y_q - y_t)^2}, \theta_q - \theta_t\right) = (d_{tq}, \theta_{tq}) = g$$

If g is within the maximum permitted distance d_{max} and angle θ_{max}, alignment is applied to every q up to the amount determined by g. If the distance and angle between $q'(q' \in Q')$ of the aligned point set Q' and closest point t of set T, are less than permitted distance (d_{err}), these points are corresponding points and allocated to set M (one query point corresponds with only one vault point and vice versa). If the size of set M exceeds the required minimum corresponding number τ, then set M is returned and if not, $\emptyset$ is returned.

In this algorithm, the maximum comparison frequency is $n \times l$. Specifically, when the size of the vault is 240 and the size of query minutiae is 40, all tests are carried out within the maximum 9,600 trials.

■ Alignment

In the absence of noise and other deformation, the rotation and displacement between two fingerprint images can be completely determined using two corresponding point pairs [18]. In an ideal scenario, the true alignment can be estimated by testing all possible pairs of minutiae for correspondence and then selecting the best correspondence [18].

Essentially, the size of the genuine set is usually different from that of the query set and selecting the overlapped set that uses only the location of minutia is computationally very expensive [18]. By using direction θ of minutia for rotation, the alignment can avoid these problems.

In our alignment step, every point of query set Q is displaced and rotated in order for a query point q_j to overlap with a target point t_i of target set T. When $q_j = (x_q, y_q, \theta_q)$, $t_i = (x_t, y_t, \theta_t)$ and the difference between two points $\Delta p = (\Delta x, \Delta y, \Delta\theta) = (x_t - x_q, y_t - y_q, \theta_t - \theta_q)$, all points of set Q are displaced by the expression, $(x + \Delta x, y + \Delta y)$. Additionally, when a displaced point $q' = (x', y', \theta')$ and the distance between q' and q_j can be expressed as $(x_r, y_r) = (x' - x_q, y' - y_q)$, then the rotated point $q'' = (x'', y'', \theta'')$ is determined as follows.

$$x'' = x_r \cdot \cos\Delta\theta - y_r \cdot \sin\Delta\theta$$
$$y'' = y_r \cdot \cos\Delta\theta + x_r \cdot \sin\Delta\theta$$
$$\theta'' = \theta_j + \Delta\theta$$

The aligned query and genuine minutiae set are illustrated in **Fig. 2**.

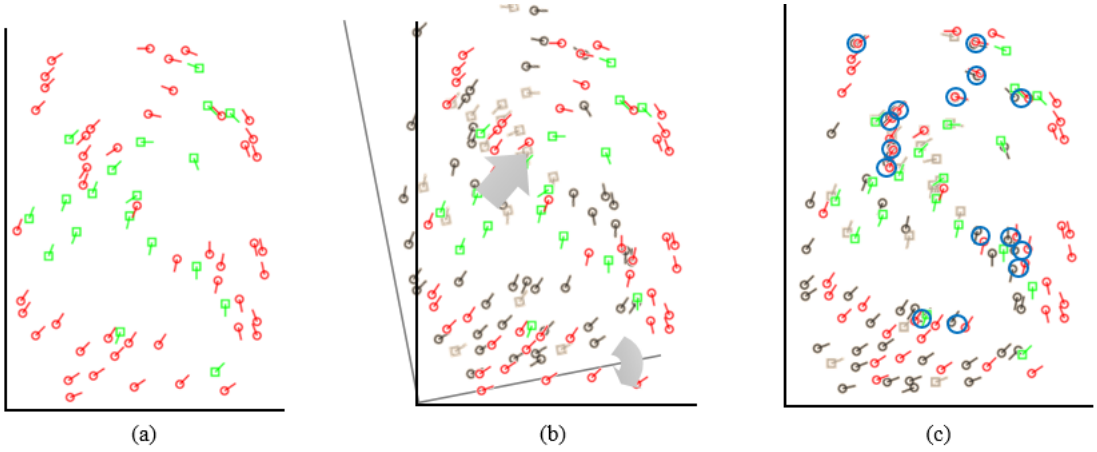

Fig. 2. Fingerprint minutiae alignment : (a) Genuine Minutiae(green points : ridge bifurcation, red points : ridge ending), (b) Genuine Minutiae and Query Minutiae overlapped(required translation and rotation, gray pattern color points : query minutiae), (c) Translated & Rotated Query Minutiae and matched points

Fig. 2 (a) depicts the genuine minutiae and **Fig. 2** (b) depicts the situation when the query minutiae for authentication are overlapped. If every point of query minutiae q corresponds to genuine minutiae t and if the query minutiae are then moved such that the position and the direction of the two points become identical, then t and q are located where most of the corresponding points are located, as shown in **Fig. 2** (c). Our matching algorithm tests all possible pairs (t, q) and selects the pair with the best correspondence.

■ Arranging the X Coordinate

Application of our matching algorithm to each point inside the vault requires the x-coordinate of the vault to include x, y, and θ of the minutiae. This condition is met by expressing these values as a 4-byte integer. On a 1000×1000 pixels XY plane, each x- and y-coordinate can be represented within 2^{10} and the direction of MINDTCT, which is less than 32, can be represented within 2^5. Thus, we can concatenate the x- and y-coordinates (10 bits each) and direction θ (5 bits) of a minutia as $[x \mid y \mid \theta]$ to arrive at the 32-bit(4-byte integer) locking/unlocking data (with the first 7 bits free).

■ Arranging Coefficients

When η is the degree of protected polynomial $\mathcal{P}$, then η is proportional to the size of secret κ; a number of selected genuine points is affected by this. Because the number of detectable minutiae is limited, it is necessary to reduce η. Reduction of η is achieved by dividing κ into non-overlapping 16-bit segments, and each segment is declared as a specific coefficient.

For instance, a 32-byte (256 bits) κ can be represented as a polynomial with 16 (256/16) coefficients in GF(2^{16}), with $\eta = 15$.

■ Generating Chaff Points

A random chaff point is computationally distinguishable because it can include unrealistic values and is vulnerable to brute force attacks [7][19]. Our chaff points are intentionally generated such that they can never be indistinguishable from genuine points.

A reserved point set *R*, as explained previously, is used as the primary restriction for creating chaff point *C*. Under the condition of the chaff point set $C = \{(\alpha_i, \beta_i)\}_{i=1}^{m}$, a 2D Boolean matrix with the same size of fingerprint images is made in order to create α_i. In the matrix, the coordinates of a reserved point set *R* and surrounding area $(x, y) \pm (d_{err} \cdot f)$ with "false" marks, are designated as a free area (Note that d_{err} is the permitted distance using our matching algorithm and f is the free area ratio). Next, a chaff point position (x_i, y_i) is randomly chosen from the region outside of the free area. This point and its associated free area are marked. The marked locations are excluded when selecting the next chaff point. Additionally, the direction of chaff point θ_i is randomly selected between integers of 0 and 31.

This step is repeated until the number of chaff points reaches the required number to fill the matrix with the free area. The generated point (x_i, y_i, θ_i) is encoded into an x-coordinate of the vault by methods used to encode the genuine point set *G*. To provide a similarity with $\mathcal{P}(\alpha_i)$, β_i is randomly generated by the same bit size with one of the y-coordinates of *G* and $\beta_i \neq \mathcal{P}(\alpha_i)$.

■ Assembly

The genuine point set *G*, chaff point set *C*, and the number of the minimum corresponding points τ are stored in a Fuzzy Vault, where τ is subject to the capacity of the fingerprint scanner and the minutiae detecting algorithm.

3.2 Authentication

The system unlocks the Fuzzy Vault with the recognized user's fingerprint minutiae, restores polynomial $\mathcal{P}$, and then extracts secret κ.

The FIDO client decrypts a user's private key by κ for FIDO authentication, signing the FIDO challenge as a private key, and finally requesting authentication to the server.

First, the query minutiae are extracted from the user's fingerprint image. Next, the x-coordinate of the vault is decoded into (x, y, θ) and overlapped points between the decoded point set and query point set are extracted using our point-to-point matching algorithm. If the number of corresponding points is less than the required correspondent number τ, then the authentication has failed. On the contrary, if the number of correspondent points is τ or more than τ, secret κ can be recovered from the corresponding

points and the recovered κ is used to decrypt the user's private key for FIDO authentication. **Fig. 3** illustrates the authentication flow of our simple Fuzzy Vault.

Our Fuzzy Vault omits any verification or error correction codes, such as the CRC code and Reed-Solomon code. The purpose of the FIDO authentication is to authenticate online; thus, whether the authentication is successful is entirely dependent on the FIDO server and the biometrics would no longer be required. The FIDO client unlocks vaults with query fingerprints and it is sufficient to decrypt the user private key by using κ. Whether this decryption is successful is determined by the FIDO server that receives a signed challenge. When the authentication on the server fails, the system either requests re-authentication from the users or locks the user's account when the maximum number of failure attempts is exceeded.

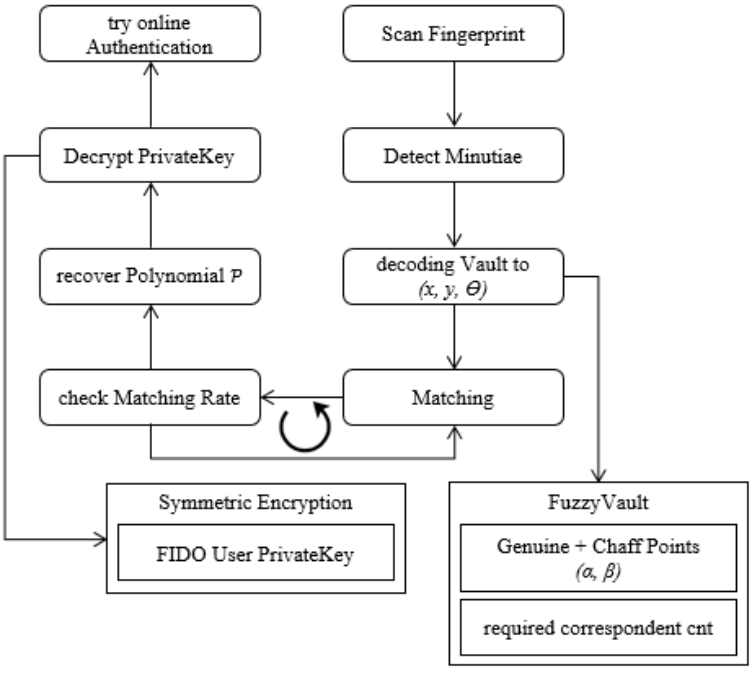

Fig. 3. Authentication flow

■ Recovering a Polynomial

In our simple Fuzzy Vault, the original polynomial $\mathcal{P}$ is reconstructed by using Lagrange interpolation methods, according to the approach followed in previous studies [9][10][14][17]. We construct a polynomial $\mathcal{P}(x) = a_\eta x^\eta + a_{\eta-1}x^{\eta-1} \ldots + a_0$ by Lagrange interpolation and concatenate its coefficients as $\kappa = a_\eta \mid a_{\eta-1} \mid \ldots \mid a_0$.

■ Error Correction

In [4], the Reed-Solomon algorithm performs decoding successfully if at least $\frac{\eta + n}{2}$ points in Q share a common polynomial (n is the number of genuine points). For example, when $\eta = 14$, $n = 22$ then $\frac{\eta + n}{2} = 18$. Computing all possible subsets required an average of 3,657.5 trials [4].

Considering translation, rotation, and distortion of a fingerprint image, the number of sharing points is nearly inconceivably high and therefore, pre-alignment of the minutiae is required. In addition, when using larger numbers of n and κ, the number of trials required to compute all possible subsets greatly increase. For instance, in our case, when $n = 40$, $\eta = 15$, then the required number of sharing points are at least 28 and the maximum number of trials is set to $\binom{40}{12} = 5{,}586{,}853{,}480$, which is unfeasible.

The CRC code, used in previous studies, provides simpler access than decoding using the Reed-Solomon algorithm. However, the CRC code provides codes for verifying a candidate coefficient. If intruders who intercept the Fuzzy Vault were to gain access to the CRC code, they could consequently locate the secret κ faster through a brute force attack under offline conditions [3][6][14]. Moreover, CRC has other flaws, some of which are presented in [15].

Our Fuzzy Vault does not execute any error correction or verification code. In order to reconstruct $\mathcal{P}$, which is $\eta + 1 = 16$, 16 sets of x-y coordinates are desired. Simply stated, preconditions to recover all genuine points are eliminated. For example, when the total number of points in the vault is 132 and the number of genuine points among them is 32 and τ is 0.5, authentication would be successful provided more than 16 genuine points could be selected. On the other hand, if more than one chaff point occurs among the selected 16 points, then the authentication has failed. Moreover, with regards to FIDO authentication, user acceptance is only available online.

In our Fuzzy Vault, when the number of overlapped points exceeds the required number of corresponding points τ, then the polynomial is reconstructed; otherwise, authentication fails. In the case of restoration of the polynomial, if the restored polynomial is unmatched, verification on the server will also be unsuccessful.

4. Security

Our Fuzzy Vault features vault excluding error correction or verification code and a free area with the identical genuine and chaff points. This feature specifies that the Fuzzy Vault is secure from extraction of the original point by free area calculation. Consequently, the security of our Fuzzy Vault is proportional to the number of chaff points. When the number of chaff points is 100 and the number of genuine points is 32, the percentage of 16-unit genuine points that can be selected is :

$$\frac{\binom{32}{16}}{\binom{132}{16}} \fallingdotseq 3.81603 \times 10^{-12} = 3.81603 \times 10^{-10}\ \%$$

When the number of chaff points is 200, the percentage becomes 3.03079×10^{-14} %. Thus, brute force attacks that select points cannot be verified offline and will be blocked online via auxiliary security systems (such as the failure count of authentication attempts).

The number of selectable chaff points in our algorithm is based on the size of the fingerprint images and the number of dots per inch(DPI). As the image size increases, the area of selectable chaff points widens, and consequently, the DPI increases, which reduces the size of the free area.

■ Brute Force Attacks

The complexity of brute force attacks grows with either the degree of the polynomial, the number of chaff points, and/or a reduction of the minutiae [7].

The quality of fingerprint images regulates the number of minutiae and the number of chaff points in our vault. The value of η depends on the length of protected secret κ and our vault supports $\eta + 1 \geq 16$. Although these features can be relatively disadvantageous to brute force attacks, our vault remains structurally safe as a result of not needing any additional code to verify an extracted polynomial.

■ Correlation Attacks

A well-known vulnerability of the Fuzzy Vault scheme is the correlation attack. For example, an attacker has intercepted two records that have been generated from the same finger. An attacker could exploit the correlation among the vaults and reveal the genuine point set [6][8][11][12][20][21]. The randomness of the chaff points and pre-alignment will allow a correlation attack, which can lead to the location of the genuine point set [7][17].

Chaff points in our Fuzzy Vault have structural similarities with the genuine point and this makes it difficult to detect a genuine point set by correlation because it does not use pre-alignment.

■ Key Inversion Attacks

If the key is known, through shoulder surfing for instance, the genuine point set in the Fuzzy Vault can be revealed easily through the restored polynomial [6][11].

In our enrollment step, κ is generated randomly, concealed by the vault, and is not stored. Thus, even though it is possible for an attacker to guess the candidates of κ, there is no verification code in the vault and thus, the attacker is unable to verify the speculative key.

5. Experimental Results

We used the open fingerprint database of NEURO Technology, which containts sufficient minutiae to test our Fuzzy Vault [22]. This database provides 76 fingerprints, 8 images per fingerprint, with each image comprising 504 × 480 pixels with a pixel density of 500 DPI. All the experiments were conducted in a Java SE Runtime Environment (version 8) and on a laptop with a 2.5 GHz Mobile CPU with 8 GB RAM.

For practical proposes, we experimented with 114 fingerprint images with ridge ending minutiae between 32 and 40 ($REL \geq 0.15$). We eliminated false minutiae that had a REL of less than 0.15 and that was unfiltered by the MINDTCT. When the matching point ratio was 50%, reconstruction of the polynomial ($\eta = 15$) and 32 genuine points were required. Conjointly, FRR is increased by the number of chaff points when there is a sizable difference between genuine minutiae and query minutiae size. Thus, we set the maximum minutiae limit to 40. These options are required to make adjustments for capacity when using the fingerprint scanner. We also adjusted various other options during the test (Free area $f: 1.5–2.0$, required correspondent ratio $\tau: 45\%–65\%$).

The result of the test is presented in **Table 1** and **Fig. 4.**

Table 1. Experimental results

MATCHING RATIO (τ)	FREE AREA (f)	GAR(%)	FAR(%)	TRIALS avg(max)	DECODING TIME(ms)	CHAFF POINTS
0.45	1.50	66.31578947	0.02399616	907.2174 (1320)	327.5272	99.99123
	1.75	74.21052632	0.01599744	794.0467 (1193)	276.0834	82.84211
	2.00	80.00000000	0.00799872	660.9138 (1060)	186.4288	61.42982
0.50	1.50	66.84210526	0.00799872	909.3254 (1378)	337.6732	100
	1.75	68.94736842	0.00799872	797.03 (1289)	277.7825	82.82456
	2.00	73.94736842	0.00799872	658.3072 (1025)	184.6612	61.08772
0.55	1.50	60.78947368	0.0	911.8416 (1350)	349.8316	99.96491
	1.75	63.94736842	0.0	797.1825 (1255)	276.3047	82.89474
	2.00	66.57894737	0.0	665.9356 (1049)	188.3832	61.67544
0.60	1.50	55.26315789	0.0	913.8952 (1391)	354.8078	100
	1.75	57.10526316	0.0	800.6623 (1258)	278.8038	83.18421
	2.00	62.63157895	0.0	666.4431 (1052)	188.5814	61.69298

	1.50	44.47368421	0.0	914.7438 (1363)	354.1693	100
	1.75	48.15789474	0.0	802.6291 (1236)	278.0598	82.92982
0.65	2.00	47.89473684	0.0	667.929 (1089)	188.1377	61.47368

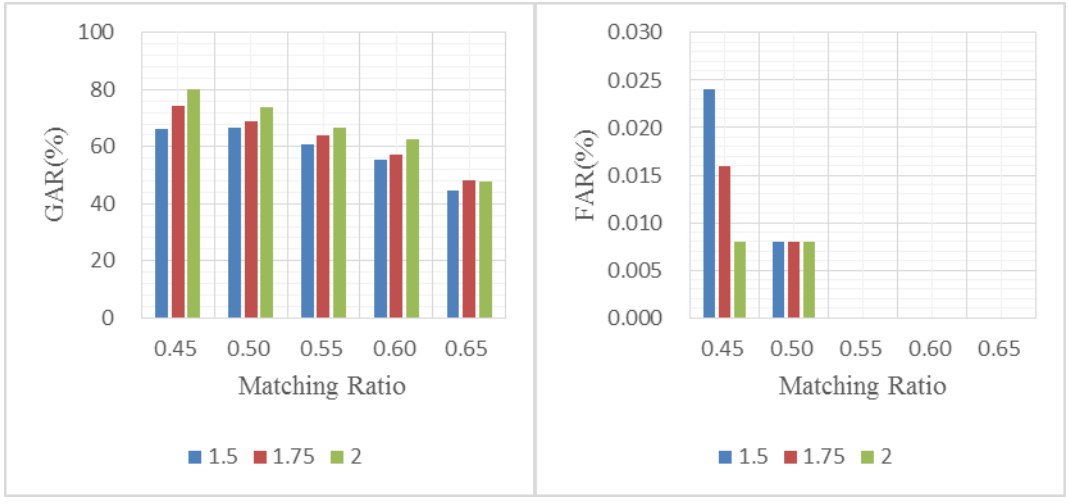

Fig. 4. GAR and FAR as a function of the matching ratio and free area

In **Table 1**, the DECODING TIME is the average time spent for each authentication and TRIALS is the average and maximum count of alignment trials per authentication. Except for the comparison with the same image, the number of same fingerprint comparisons is 380 and the number of different fingerprint comparisons is 12,502.

In our experimental results, the GAR is recorded from 47.89% to 80.00% and the FAR is recorded from 0.02% to 0%. All corresponding combinations had an average of 791 trials. The average value of the DECODING TIME was 270 ms. The number of generated chaff points was 61 to 100. For example, if τ and f are 0.50 and 1.5, respectively, and the number of genuine and chaff points in the vault were 32 and 100, respectively, the total number of possible combinations is $\binom{132}{16}$ and among these combinations, $\binom{32}{16}$ combinations will successfully decode the secret. The probability of a combination of points decoding the secret $\binom{32}{16} \Big/ \binom{132}{16}$, and the expected number of combinations that need to be evaluated is

2.6×10^{11}. This corresponds to a computational time of 2,805.94 years based on our current implementation. In the worst case, if τ and f are 0.45 and 2.0, respectively, and the number of genuine and chaff points are 32 and 61, respectively, the expected number of

combinations is 6.3×10^8, which corresponds to 3.71 years. With respect to FIDO features that should not transmit biometric templates over the network, the FIDO client can choose this option in order to increase GAR.

Table 2 compares the results obtained with our implementation and those from other similar works in terms of the GAR, FAR, decoding time, security, and error correction algorithm.

Table 2. Comparison with other similar works

Works	η	DB	Chaff points	GAR (%)	FAR (%)	Decoding Time	Security	Verifying Code
[5]	9	FVC2002-DB2	vault : 250 – 300	90 – 83	0	-	-	Hashing
[8]	15	MCYT-Fingerprint-100	200	42.72	0	3.2GHz 772 ms	-	Reed-Solomon
[9]	8	IBM-GTDB	200	79	0	3.4 GHz 52 s	C(218, 9)/C(18,9) 439 years	CRC
[10]	7 – 10	FVC2002-DB2	200 – 206	91 – 86	0.13 – 0	3.4 GHz 8 s	C(224, 9)/C(24, 9) 13 years	CRC
[14]	8 – 11	FVC2002-DB1	vault : 200	92 – 74	13 – 0	-	-	Hashing
[23]	13	FVC2002-DB2	400 – 414	92	0	2.33GHz 192 ms	47-bit (660,488 years) – 32 bit (20 years)	Hashing
Proposed (0.45 – 0.55)	15	NEURO	100 – 61	80 – 61	0.02 – 0	2.5 GHz M 267 ms	C(132, 16)/C(32/16) 2,805.94 years -	-

							C(93, 16)/C(32/16) 3.71 years	

In **Table 2**, η is the degree of the polynomial applied in the evaluation test of the method from each previous study. The chaff points are either the total size of the vault or the number of chaff points inside the vault. Security is recorded on the basis of the content reported from each study. Because of different testing environments across these studies, the above number is not the absolute standard.

As shown in **Table 2**, our algorithm supports a higher value for η than other works and provides enhanced security with a smaller number of chaff points. Additionally, our algorithm, when compared to the other studies, has a moderate value for the GAR, a low value for the FAR, and a shorter decoding time. Verification algorithms, such as hash, CRC, and the Reed-Solomon code, are used to verify whether query minutiae have security problems. Our study is different from other relevant studies because we do not use a verification algorithm; therefore, our solution is not adversely affected by problems such as those mentioned above. This feature shows that our algorithm is suitable for the mechanism wherein fingerprints are processed in FIDO clients.

6. Conclusion

In this paper, we proposed a simple Fuzzy Vault that can be used in FIDO authentication. Our point-to-point matching algorithm tests all possible pairs for correspondence between the point of query fingerprint minutiae and the point of vault while using (x, y, θ) and selects the best correspondence and reconstructs the polynomial P. The proposed Fuzzy Vault uniquely differs from methods proposed in previous studies, such as Reed-Solomon, CRC, and hashcode, because it excludes the use of error correction and verification code. Because this work was carried out in tandem with FIDO authentication, our Fuzzy Vault has the ability to prevent brute force attacks. During the enrollment process, our Fuzzy Vault randomly creates a secret, which is not stored, and uses a chaff point generation algorithm. The careful consideration and combination of these features enables our simple Fuzzy Vault to evade threats of correlation and key inversion attacks.

We performed a trial using an open fingerprint database to verify the effectiveness of our Fuzzy Vault. The experimental results were as follows: the GAR was recorded from 47.89% to 80.00% and the FAR was recorded from 0.02% to 0% with adjusting options.

The current implementation of our Fuzzy Vault only uses ridge ending among fingerprint minutiae. This limitation can be remedied by lowering the matching accuracy; therefore both ridge ending and ridge bifurcation can be utilized in future studies. Furthermore, we used MINDTCT of NIST for detecting the minutiae, which treats 11.25 degrees as the unit for DIR[16]. This can be resolved by lowering our matching algorithm to use a smaller

degree value, which would result in a higher GAR.

References

[1] HyunJin Kim, JunHoo Park, JangYong Lee and JaeCheol Ryou, "Biometric authentication technology trends in smart device environment," in *Proc. of Mobile and Wireless Technology 2015*, vol. 310, pp. 199-206, 2015. Article (CrossRef Link).

[2] fido alliance, https://fidoalliance.org/

[3] B. Tams, P. Mihăilescu and A. Munk, "Security Considerations in Minutiae-Based Fuzzy Vaults," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 5, pp. 985-998, May. 2015. Article (CrossRef Link).

[4] Ari Juels and Madhu Sudan, "A Fuzzy Vault Scheme," *IEEE International Symposium on Information Theory*, pp. 408-426, 2002. Article (CrossRef Link).

[5] Örencik, C., Pedersen, T. B., Savaş, E. and Keskinöz, M, "Securing fuzzy vault schemes through biometric hashing," in *Proc. of Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 18, no. 4, pp. 515-540, 2010. Article (CrossRef Link).

[6] Walter J. Scheirer and Terrance E. Boult, " Cracking Fuzzy Vaults and Biometric Encryption," in *Proc. of Biometrics Symposium*, pp. 1-6, Sep. 2007. Article (CrossRef Link).

[7] P. Mihailescu, "The fuzzy vault for fingerprints is vulnerable to brute force attack," in *Proc. of Computer Vision and Pattern Recognition*, Oct. 2007. Article (CrossRef Link).

[8] Benjamin Tams, "Unlinkable Minutiae-Based Fuzzy Vault," in *Proc. of IET Biometric*, pp. 11-39, Jun. 2015. Article (CrossRef Link).

[9] Umut Uludag, Sharath Pankanti and Anil K. Jain, "Fuzzy Vault for Fingerprints," in *Proc. of Audio- and Video-Based Biometric Person Authentication*, vol. 3546, pp. 310-319, 2005. Article (CrossRef Link).

[10] Karthik Nandakumar, Anil K. Jain and Sharath Pankanti, "Fingerprint-Based Fuzzy Vault : Implementation and Performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744-757, Dec. 2007. Article (CrossRef Link).

[11] Lu Leng and Andrew Beng Jin Teoh, "Alignment-free row-co-occurrence cancelable palmprint Fuzzy Vault," in *Proc. of Pattern Recognition*, vol. 48, no. 7, Jul. 2015. Article (CrossRef Link).

[12] Woo Yong Choi, Yongwha Chung, Jin-Won Park and Dowon Hong, "Fingerprint Template Protection Using One-Time Fuzzy Vault," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 5, no. 11, pp. 2221-2234, Nov. 2011. Article (CrossRef Link).

[13] Qiong Li, Zhaoqing Liu and Xiamu Niu, "Analysis and Problems on Fuzzy Vault Scheme," in *Proc. of Intelligent Information Hiding and Multimedia Signal Processing*, pp. 244-250, Dec. 2006. Article (CrossRef Link).

[14] Minh Tan Nguyen, Quang Hai Truong and Tran Khanh Dang, "Enhance fuzzy vault security using nonrandom chaff point generator," in *Proc. of Information Processing Letters*, vol. 116, no. 1, pp. 53-64, Jan.2016. Article (CrossRef Link).

[15] Hoi Ting Poon and Ali Miri, "On Efficient Decoding for the Fuzzy Vault scheme," in *Proc. of Information Science, Signal Processing and their Applications (ISSPA)*, pp. 454-459, Jul. 2012. Article (CrossRef Link).

[16] User's Guide to NIST Biometric Image Software (NBIS), National Institute of Standards and Technology, 2015. Article (CrossRef Link).

[17] Cai Li and Jiankun Hu, "A Security-Enhanced Alignment-Free Fuzzy Vault-Based Fingerprint Cryptosystem Using Pair-Polar Minutiae Structures," *IEEE Transactions on Information*

Forensics and Security, vol. 11, no. 3, pp. 543-555, Dec. 2015. Article (CrossRef Link).
[18] Karthik Nandakumar and Anil K. Jain, "Local Correlation-based Fingerprint Matching," in *Proc. of ICVGIP 2004*, pp. 503-508, Dec. 2004. Article (CrossRef Link).
[19] E.C. Chang, R. Shen and F.W. Teo, "Finding the original point set hidden among chaff," in *Proc. of the 2006 ACM Symposium on Information, Computer and Communications Security*, pp. 182–188, Mar. 2006. Article (CrossRef Link).
[20] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," in *Proc. of SPIE, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819, pp. 1–7, 2008. Article (CrossRef Link).
[21] Hoi Ting Poon and Ali Miri, "A collusion attack on the fuzzy vault scheme," in *Proc. of The ISC, Int. J. Inf. Secur. 1*, vol. 1, no. 1, pp. 27–34, Jan. 2009. Article (CrossRef Link).
[22] Neuro Technology, Article (CrossRef Link).
[23] Peng Li, Xin Yang, Kai Cao, Xunqiang Tao, Ruifang Wang and Jie Tian, "An alignment-free fingerprint cryptosystem based on fuzzy vault scheme," in *Proc. of Journal of Network and Computer Applications 2010*, vol. 33, no. 3, pp. 207-220, May. 2010. Article (CrossRef Link).

Dongil Cho received M.S. and Ph.D. degrees in software engineering from Soongsil University, Seoul, Korea on 2008, and 2012. I am working for Tomato System Korea from 2003 as a researcher. My research interests include Web Security, Web Middleware Engineering, and Web UI.