Secure Key Agreement using Pure Biometrics

Dilara Akdoğan
Sabancı University
İstanbul, TURKEY
Email: dakdogan@sabanciuniv.edu

Duygu Karaoğlan Altop
Sabancı University
İstanbul, TURKEY
Email: duyguk@sabanciuniv.edu

Albert Levi
Sabancı University
İstanbul, TURKEY
Email: levi@sabanciuniv.edu

***Abstract*—In this paper, we propose a novel secure key agreement protocol that uses biometrics with unordered set of features. Our protocol enables the user and the server to agree on a symmetric key, which is generated by utilizing only the feature points of the user's biometrics. It means that our protocol does not generate the key randomly or it does not use any random data in the key itself. As a proof of concept, we instantiate our protocol model using fingerprints. In our protocol, we employ a threshold-based quantization mechanism, in order to group the minutiae in a predefined neighborhood. In this way, we increase the chance of user-server agreement on the same set of minutiae. Our protocol works in rounds. In each round, depending on the calculated similarity score on the common set of minutiae, the acceptance/rejection decision is made. Besides, we employ multi-criteria security analyses for our proposed protocol. These security analyses show that the generated keys possess acceptable randomness according to Shannon's entropy. In addition, the keys, which are generated after each protocol run, are indistinguishable from each other, as measured by the Hamming distance metric. Our protocol is also robust against brute-force, replay and impersonation attacks, proven by high attack complexity and low equal error rates.**

I. Introduction

In generic cryptographic applications, unique and user-specific secret keys are used. However, these keys can be stolen, lost or willingly shared. On the other hand, biometric traits are used to identify or verify users since they are strictly bound to the user. The reason behind this is that the biometric traits are unique physical, physiological or behavioral characteristics of individuals. In order to provide higher security and privacy, biometrics and cryptography are combined as this combination provides the binding of user's personal characteristics to cryptographic keys. The combination of biometrics and cryptography is referred to as *crypto-biometric* or *bio-cryptographic systems*. This way, the secret keys, which are used for encryption and decryption in cryptographic applications, are derived from the biometric data with the help of their unique features.

Each biometric data has its own distinctive features. These features can be represented with either ordered or unordered sets. Ordered sets are typically binary strings. Iris is an example of a biometric with ordered features, as iris code is a binary string retrieved from the unrolled iris texture. On the other hand, unordered sets are generally a list of points on a coordinate system with insignificant orders. Biometric features from which independent points can be extracted are examples of unordered feature sets, such as fingerprints.

In this paper, we propose a novel biometric key agreement protocol that uses unordered set of features. As an example, we implement and evaluate our protocol using fingerprints, which are represented with unordered set of minutiae points. Our protocol generates the keys by using only the minutiae points, without any other helper component. In this key agreement protocol, hash functions and threshold mechanisms are employed. Moreover, in order to mask the genuine minutiae points, fake minutiae points are generated according to a strategy. This strategy is developed to properly manage the trade-off between information leakage to the attacker and acceptable verification results. For this reason, a distance threshold and a neighborhood relation are defined such that there cannot be more than one point (genuine and/or fake) in a pre-defined distance neighborhood. This process is analogous to quantization.

Our key agreement protocol runs in a round-manner such that at each round, the user and the server tries to find a common set of minutiae points. At the end of the protocol, either the user is rejected due to the reason that the similarity score is below the acceptance threshold or the user and the server agree on a secure symmetric key.

We analyzed the security performance of our system from different perspectives. From biometrics point of view, our model shows high verification performance, proven by low Equal Error Rates (EER). We also analyzed the resistance of our protocol against some known attacks, such as brute-force, replay and impersonation attacks. Moreover, the quality of the agreed keys is analyzed in terms of randomness and distinctiveness. These analyses show that our system is quite resistant to these attacks; the generated keys are random enough to be used as cryptographic keys; and each key is distinct from the other agreed keys.

The rest of this paper is organized as follows. Section II gives background information and summarizes the related work in the literature. In Section III, we introduce our proposed secure key agreement protocol. Section IV evaluates the performance of our proposed protocol and discusses its security analyses. Finally, in Section V we conclude the paper and provide some future works.

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 114E557.

Dilara Akdoğan was supported by TÜBİTAK BİDEB 2228.

Duygu Karaoğlan Altop was supported by TÜBİTAK BİDEB 2211-C.

II. Related Work

Bio-cryptographic systems are threefold: (i) key release, (ii) key generation and (iii) key binding. In [1] and [2], the authors describe these bio-cryptographic methods and discuss their problems. In the key release mechanism, firstly a biometric authentication process is run. In this process, the input biometric template is compared with the one in the system database. If the matching is successful, a key is released. However, the fact that the user authentication and key release are independent processes is a disadvantage of this mechanism. The key and the user biometric are not strictly bound to each other. On the other hand, in key generation or key binding mechanisms, biometrics and cryptography are integrated; a cryptographic key is bound to the biometric data of the user. In these methods, neither the biometric template, nor the cryptographic key is accessible to the attacker. The correct cryptographic key could only be generated when a valid biometric template is presented by the user. Biometric matching is not performed, because when the correct key cannot be generated from the biometric template, the decryption function fails and the user is rejected automatically. The reason behind this is that cryptographic encryption/decryption functions require exactly the same key.

The main issue in biometric key generation or key binding methods is the variance of biometric template. Due to variations in biometrics, if one bit of the generated key is different than that of the correct one, the genuine user may be rejected. In order to avoid these false rejects, fuzzy key binding methods are proposed, namely *fuzzy commitment* and *fuzzy vault*. The *fuzzy commitment* scheme was proposed by Juels and Wattenberg [3]. In this scheme, the user selects a secret word W. The difference (XOR) between the user's biometric template X and the codeword W is denoted as d. The difference vector d and the hash of secret word $y = H(W)$ together constitute the encrypted message. At the verification stage, the user provides a query biometric Y. The difference between Y and d is $W^{'}$, and y is used to check the correctness of the extracted $W^{'}$. This scheme is applicable to be used with ordered set of features, such as iris code. In [4], the authors propose a mechanism to obtain cryptographic keys from iris codes using fuzzy commitment, which is based on the method described in [5]. However, [6] and [7] shows that the fuzzy commitment methods are vulnerable, because the attacker can reconstruct not only the key but also the iris code by making use of the error correction codes and statistical attacks.

The *fuzzy vault* scheme is proposed by Juels and Sudan [8]. In contrast to the fuzzy commitment scheme, the fuzzy vault scheme is applied to the unordered set of features, such as minutiae in fingerprints. In this method, a secret word W is mapped to the coefficients of a polynomial $P(x)$. This polynomial is evaluated on the feature points of the biometric template. In addition to the evaluated points $(x, P(x))$, a large number of chaff points that do not lie on the polynomial are generated. These genuine and chaff points are mixed and the total set is named as the *vault*. In the verification stage, the user provides a query template, and with the help of this template, genuine points are determined. Using Lagrange Interpolation, the polynomial $P(x)$ is reconstructed and its coefficients are mapped to the secret $W^{'}$. With the use of error correction codes, the correct secret W is obtained. In [9] and [10], fingerprint-based fuzzy vault methods are presented. However, it is proven that the fuzzy vault is also vulnerable to some known attacks; such as brute-force [11], stolen-key inversion [12] and correlation based attacks [13]. Although in [14], the authors improve fuzzy vault for fingerprint verification, they still leak some information to the attacker by inserting chaff points into the vault that are close to each other but away from the genuine points. With this strategy, the attacker can make sure that if there are two points which are close enough to each other and if it is known that one of these points is chaff, the other point is definitely a chaff point.

In both of the fuzzy commitment and the fuzzy vault mechanisms, the secret word is selected by the user or it is randomly generated. In other words, the secret word is not derived directly from the biometric template. On the contrary, the biometric template is used as a component to hide the secret word. In contrast to all of these methods, in this paper, the secret word (key) is directly generated from the feature points of the biometric template.

III. Proposed Secure Key Agreement Protocol using Pure Biometrics

In this section we describe our proposed secure key agreement protocol, which uses fingerprint biometrics without any other type of helper data. The definitions of the symbols used in the protocol definition are given in Table I. Our key agreement protocol can be divided into two phases: (i) enrollment and (ii) verification, each of which is explained in the following subsections.

A. Enrollment Phase

The enrollment is performed only at the server side and the corresponding template generation algorithm is given in Algorithm 1. At the enrollment stage, the user provides three fingerprint images, *FP*$_1$, *FP*$_2$, *FP*$_3$, of the same finger. Then, the minutiae of these fingerprints are extracted. Each minutia is represented with three attributes: x-coordinate, y-coordinate and *type*. The type of a minutia can be *end* or *bifurcation*. *End* type of a minutia indicates a ridge ending. On the other hand, if the ridge branches into two, the branching point is a *bifurcation* type of a minutia. The minutiae list of a fingerprint image constitutes the template of this particular fingerprint image. While generating the template, we quantize the minutiae by selecting representatives from the groups that are determined by the predefined distance threshold, T_{dist}. In this quantization step, the minutiae which are at most T_{dist}-away to any other minutia are mapped to one minutia by picking the one with the smallest y-coordinate value. After that, the server puts these fingerprint templates on top of each other, in order to find out the most reliable minutiae.

TABLE I
SYMBOLS USED IN PROTOCOL DEFINITION

Symbol		***Description***
FP		Fingerprint
x		x-coordinate of a minutia
y		y-coordinate of a minutia
type		Type of a minutia
n_u		Total number of genuine minutiae on the user side
n_s		Total number of genuine minutiae on the server side
n_{com}		Number of common minutiae found by the server
n_{com}^{key}		Number of minutiae used in the final key agreement
$H^i(\cdot)$		Hash function applied i times ($i \geq 0$)
G_s		Set of genuine minutiae on the server side
G_u		Set of genuine minutiae on the user side
C		Set of fake minutiae on the user side
Q_u		Set of shuffled $\left(H^2(g_u) \cup H^2(c)\right)$ s.t. $g_u \in G_u$ & $c \in C$
G'_s		Set of minutiae $\in \{Q_u \cap G_s\}$
$G''_{s,j}$		Any subset of G'_s s.t. $\lvert G''_{s,j}\rvert = \lvert G'_s\rvert - j \ \ (j \geq 1)$
G'_u		Any subset of G_u s.t. $\lvert G'_u\rvert = \lvert G'_s\rvert$
$G''_{u,j}$		Any subset of G'_u s.t. $\lvert G''_{u,j}\rvert = \lvert G'_s\rvert - j \ \ (j \geq 1)$
S		Similarity score
T_{sim}		Acceptance similarity score threshold
T_{dist}		Distance threshold used in neighborhood definition
$K^i_{(us,su)}$	K^i	i^{th} key generated ($i \geq 0$)
	us	by the user to communicate with the server
	su	by the server to communicate with the user
$HMAC(\cdot)$		Keyed-Hashing for Message Authentication [15]
$HMAC_{K_{us}}(\cdot)$		*HMAC* generated using K_{us}
$HMAC_{K_{su}}(\cdot)$		*HMAC* generated using K_{su}
$HMAC_{K^i_{su}}(\cdot)$		*HMAC* generated using K^i_{su}
att_c		Attack complexity

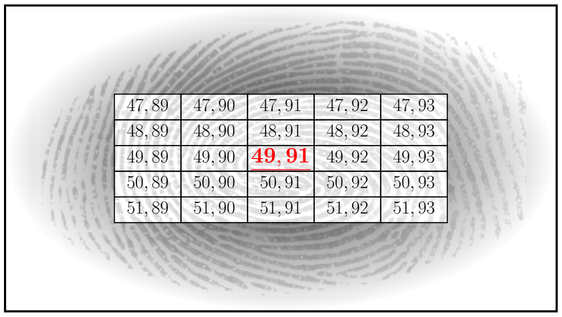

Fig. 1. Neighborhood relation when $T_{dist} = 2$

Algorithm 1 Template Generation Algorithm

```
INPUT: $FP_1,\ FP_2,\ FP_3$
OUTPUT: $G_s$
 1: $G_s^{init}$ = ExtractMinutiae($FP_1$, $FP_2$, $FP_3$)
 2: for $i = 1 : |G_s^{init}| - 1$ do
 3:     $m_1 = G_s^{init}(i)$
 4:     for $j = i + 1 : |G_s^{init}|$ do
 5:         $m_2 = G_s^{init}(j)$
 6:         if $m_1.x \geq m_2.x - (2 * T_{dist})$ &
 7:            $m_1.x \leq m_2.x + (2 * T_{dist})$ &
 8:            $m_1.y \geq m_2.y - (2 * T_{dist})$ &
 9:            $m_1.y \leq m_2.y + (2 * T_{dist})$ &
10:            $m_1.type == m_2.type$ then
11:             $m_1.visited++$
12:         end if
13:     end for
14: end for
15: for $i = 1 : |G_s^{init}|$ do
16:     if $G_s^{init}(i).visited < 2$ then
17:         Remove $i^{th}$ minutia from $G_s^{init}$
18:     end if
19: end for
20: $ind \leftarrow 1$
21: for $i = 1 : |G_s^{init}|$ do
22:     $m_1 = G_s^{init}(i)$
23:     for $j = (-1) * T_{dist} : T_{dist}$ do
24:         for $k = (-1) * T_{dist} : T_{dist}$ do
25:             $G_s(ind) = H^1(m_1.x + j || m_1.y + k || m_1.type)$
26:             $ind \leftarrow ind + 1$
27:         end for
28:     end for
29: end for
```

The minutiae which are present in at least two out of three fingerprint templates are considered as reliable minutiae. Only the reliable minutiae are kept in the final template. For the reason that the minutiae are close to each other more than T_{dist} are considered as one minutia, a T_{dist}-*neighborhood* relation is defined as follows: All of the points in the coordinate system which have x-coordinate in $[x_j - T_{dist},\ x_j + T_{dist}]$ and y-coordinate in $[y_j - T_{dist},\ y_j + T_{dist}]$ are the neighbors of the minutia with $(x_j,\ y_j)$ in the T_{dist}-*neighborhood*. This neighborhood relation is exemplified in Figure 1, where the original minutia is located at (49, 91) and T_{dist} is 2.

Thereafter, x-coordinate, y-coordinate and *type* of each minutia point and its neighbors in T_{dist}-neighborhood together with the *type* of this particular minutia are concatenated and hashed one by one as $H^1(x||y||type)$. These hashes constitute a particular user's template in the server. Note that although double hashes will be needed in the verification stage, storing single hashes is enough, since the double hashes can be calculated by re-hashing the stored values once again if necessary.

B. Verification Phase

At the verification stage, three different fingerprint images of the same finger are used. As in the enrollment phase, the minutiae points are extracted from these fingerprints. Similarly, at most T_{dist}-away minutiae are mapped to one minutia by selecting the one with the smallest y-coordinate value. After that, three fingerprint templates are put on top of each other and the most reliable minutiae are selected. In order to mask the genuine minutiae points at the user side, $(10 \times |G_u|)$ fake minutiae points are generated randomly. Fake minutiae point generation is an important process, since the fake points should not leak any information to the attacker. For this reason, a fake point must be indistinguishable from a genuine minutia point from an attacker's point of view. Since we make sure that all of the genuine minutiae points are at least T_{dist}-away from each other, fake minutiae points must be T_{dist}-away from all the other points as well. Therefore, the fake minutiae points must also preserve the T_{dist}-neighborhood relation.

After the fake minutiae point generation process ends, each minutia point's (genuine and fake) x-coordinate, y-coordinate and *type* are concatenated. Each value is double hashed as follows: $H^2(x||y||type)$. As the key will be generated using single hashed values of the genuine minutiae, the user keeps $H^1(x||y||type)$ only for the genuine minutiae. Note that in contrast to the enrollment phase, the points in the T_{dist}-neighborhood are neither hashed nor sent to the server.

The protocol flow can be seen in Figure 2. Double hashed points' list together with the ID of the user is transmitted to the server. In order to extract the genuine points from the list, the server compares each point with this particular user's double hashed template. Since the server has the neighbor minutiae points as well, if a genuine minutia of the user is in the T_{dist}-neighborhood with a minutia in the server side, it is counted as a common genuine minutia. However, it may or may not be a genuine minutia. Our protocol provides solutions in the following steps for the cases that fake minutiae are considered as genuine minutiae.

After the comparison is completed, a similarity score is calculated. There are two well-known methods to calculate the similarity score of two fingerprints as given in Equation 1 and Equation 2, where n_{com} is the number of common minutiae, n_u is the number of genuine minutiae on the user side, n_s is the number of genuine minutiae on the server side. Although, in [16], it has been claimed that Equation 2 provides better verification results, in our tests we get better results with Equation 1.

$$S = \frac{n_{com}^2}{n_u \times n_s} \times 100 \tag{1}$$

$$S = \frac{2 \times n_{com}}{n_u + n_s} \times 100 \tag{2}$$

If the calculated score is above a certain acceptance threshold, T_{sim}, the user is accepted and the key agreement process starts. In the key agreement process, the server concatenates single hashes of all common minutiae, $H^1(g'_{s,i})$, and everything is rehashed to generate K_{su}, which is the key to be used while communicating with the user. In order to make sure that the user will generate the same key, the server computes the HMAC of a predefined message msg using K_{su} and transmits this value together with the number of common found minutiae, $|G'_s|$, to the user.

Upon receiving the message, the user generates a key using one of the possible subsets of the genuine minutiae whose size is the same as the number of found minutiae on the server side. If the user can verify the HMAC using this generated key, (s)he sends a positive acknowledgment to the server. Otherwise, the user generates another key using another subset, until either the HMAC is verified or all possible subsets are exhausted. In the case that the HMAC is not verified, *RETRY* message is transmitted to the server.

If the protocol continues with the *RETRY* message, the server computes the similarity score using $|G'_s|-1$ as the number of common minutiae. If the score is above the acceptance threshold T_{sim}, the server generates all possible keys using all possible subsets of the found minutiae, whose size is equal to $|G'_s| - 1$. The server then transmits all of the HMAC values generated using these keys to the user. If the user can verify any one of these HMAC values using any one of the keys generated with any possible subset of the genuine minutiae, whose size is equal to $|G'_s| - 1$, the user transmits a positive acknowledgment and the index, *i*, of the verified HMAC to the server. Otherwise, another *RETRY* message is transmitted to the server. In this case, the same process with $|G'_s| - 2$ is carried out. The protocol stops at the j^{th} step, if either the similarity score computed using the number $|G'_s| - j$ is less than the acceptance threshold, or any HMAC value is verified by the user. If any HMAC value is verified at the end of this protocol, the server and the user can agree on a symmetric cryptographic key without using any non-biometric or random value. On the other hand, if the protocol stops without generating a symmetric key, it can start from scratch upon request.

IV. Performance and Security Analysis

Our proposed protocol is tested with 30 subjects from Verifinger Sample Database [17], which includes fingerprints scanned using Cross Match Verifier 300 at 500 ppi [18]. Each subject has 8 fingerprint images. These fingerprint images are aligned using their intensity values in MATLAB R2014b. The minutiae of each fingerprint is extracted using the Neurotechnology Biometric SDK 5.0 Verifinger [17]. First 3 fingerprint images are used to generate the template on the server side, while the remaining 5 fingerprint images are used as combinations of 3 at the user side. Hence, each subject is tested $\binom{5}{3} = 10$ times. In addition to the genuine tests, impostor tests are also carried out. In these impostor tests, each subject's template is tested against all other subjects' queries. The hash function used in the protocol is SHA-256 [19]; hence all of the generated keys are 256 bits long.

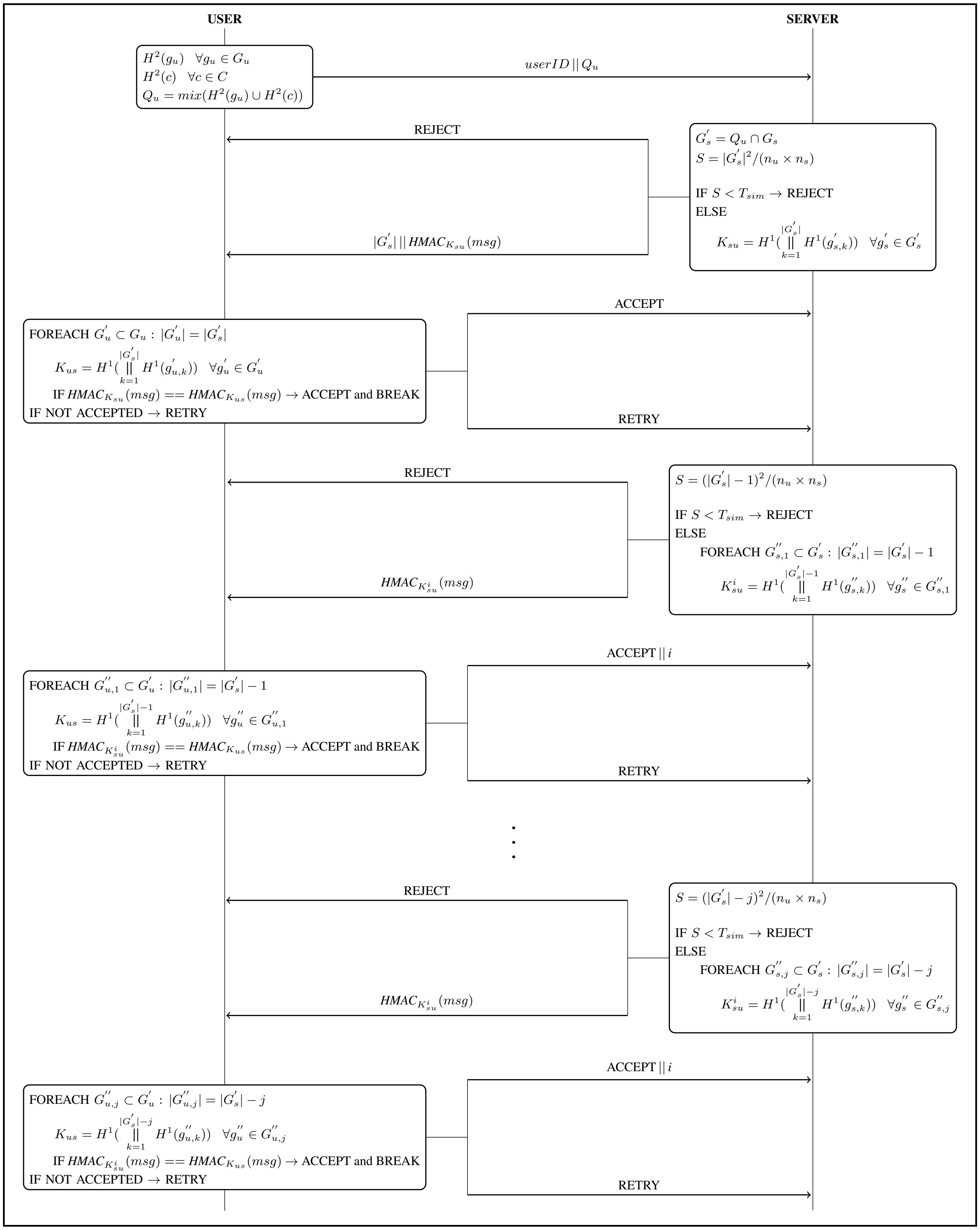
USER
SERVER
$H^2(g_u) \quad \forall g_u \in G_u$
$H^2(c) \quad \forall c \in C$
$Q_u = mix(H^2(g_u) \cup H^2(c))$
$userID \,||\, Q_u$
$G_s^{'} = Q_u \cap G_s$
$S = |G_s^{'}|^2/(n_u \times n_s)$
IF $S < T_{sim} \rightarrow$ REJECT
ELSE
$K_{su} = H^1(\mathop{||}_{k=1}^{|G_s^{'}|} H^1(g_{s,k}^{'})) \quad \forall g_s^{'} \in G_s^{'}$
REJECT
$|G_s^{'}| \,||\, \mathit{HMAC}_{K_{su}}(msg)$
FOREACH $G_u^{'} \subset G_u : |G_u^{'}| = |G_s^{'}|$
$K_{us} = H^1(\mathop{||}_{k=1}^{|G_s^{'}|} H^1(g_{u,k}^{'})) \quad \forall g_u^{'} \in G_u^{'}$
IF $\mathit{HMAC}_{K_{su}}(msg) == \mathit{HMAC}_{K_{us}}(msg) \rightarrow$ ACCEPT and BREAK
IF NOT ACCEPTED $\rightarrow$ RETRY
ACCEPT
RETRY
$S = (|G_s^{'}| - 1)^2/(n_u \times n_s)$
IF $S < T_{sim} \rightarrow$ REJECT
ELSE
FOREACH $G_{s,1}^{''} \subset G_s^{'} : |G_{s,1}^{''}| = |G_s^{'}| - 1$
$K_{su}^i = H^1(\mathop{||}_{k=1}^{|G_s^{'}|-1} H^1(g_{s,k}^{''})) \quad \forall g_s^{''} \in G_{s,1}^{''}$
REJECT
$\mathit{HMAC}_{K_{su}^i}(msg)$
FOREACH $G_{u,1}^{''} \subset G_u^{'} : |G_{u,1}^{''}| = |G_s^{'}| - 1$
$K_{us} = H^1(\mathop{||}_{k=1}^{|G_s^{'}|-1} H^1(g_{u,k}^{''})) \quad \forall g_u^{''} \in G_{u,1}^{''}$
IF $\mathit{HMAC}_{K_{su}^i}(msg) == \mathit{HMAC}_{K_{us}}(msg) \rightarrow$ ACCEPT and BREAK
IF NOT ACCEPTED $\rightarrow$ RETRY
ACCEPT $||\, i$
RETRY
$S = (|G_s^{'}| - j)^2/(n_u \times n_s)$
IF $S < T_{sim} \rightarrow$ REJECT
ELSE
FOREACH $G_{s,j}^{''} \subset G_s^{'} : |G_{s,j}^{''}| = |G_s^{'}| - j$
$K_{su}^i = H^1(\mathop{||}_{k=1}^{|G_s^{'}|-j} H^1(g_{s,k}^{''})) \quad \forall g_s^{''} \in G_{s,j}^{''}$
REJECT
$\mathit{HMAC}_{K_{su}^i}(msg)$
FOREACH $G_{u,j}^{''} \subset G_u^{'} : |G_{u,j}^{''}| = |G_s^{'}| - j$
$K_{us} = H^1(\mathop{||}_{k=1}^{|G_s^{'}|-j} H^1(g_{u,k}^{''})) \quad \forall g_u^{''} \in G_{u,j}^{''}$
IF $\mathit{HMAC}_{K_{su}^i}(msg) == \mathit{HMAC}_{K_{us}}(msg) \rightarrow$ ACCEPT and BREAK
IF NOT ACCEPTED $\rightarrow$ RETRY
ACCEPT $||\, i$
RETRY

Fig. 2. Our proposed secure key agreement protocol

In the below subsections, we discuss the verification results of the system and provide the security analysis of the protocol, as well as the randomness and distinctiveness analyses of the generated keys.

A. Verification Results

For each test, a similarity score is calculated as given in Equation 1 and Equation 2. The minimum score, the maximum score and the average score of the system are calculated for each subject. These scores are used as acceptance thresholds of the system one by one. For each different threshold, the corresponding False Accept Rate (FAR) and False Reject Rate (FRR) values of the system are calculated. FAR is the percentage of the impostor subjects who are accepted as genuine users; whereas FRR is the percentage of the genuine subjects who are rejected. As a result of these operations, the best Equal Error Rate (EER; the point where FAR = FRR) percentages are obtained when the maximum scores of the system is picked as the acceptance threshold. It is inevitable that an impostor subject cannot reach the maximum similarity score of a genuine subject. Figure 3 shows the FAR and FRR percentages when the maximum of the similarity scores, which are calculated using Equation 1, is picked as the acceptance threshold. As can be seen in this figure, our protocol achieves 0.57% EER when the optimum acceptance threshold is 5.99. This threshold value is the point where FAR and FRR curves intersect with each other. In order to minimize both FAR and FRR at the same time, their intersection point is considered. If the acceptance threshold is selected as the average score of the system, the EER lies at 5% with 2.15 acceptance threshold, as can be seen in Figure 4. On the other hand, if the acceptance threshold is selected as the minimum score of the system as in Figure 5, the EER is 10% and the acceptance threshold is 0.76. In Table II, all of these EER values that are calculated according to two different predefined similarity score equations are given. The explained results are summarized in this table as well.

TABLE II
EER VALUES

Equation	*Strategy*	*EER* (%)
Equation 1	*min*	10
	max	0.57
	avg	5
Equation 2	*min*	10
	max	0.57
	avg	6.66

B. Security Analysis

In this section, firstly our threat model is given. After that, the strength of our protocol against brute-force, replay and impersonation attacks are analyzed. In addition, the quality of the generated keys are also examined via entropy and Hamming distance analyses.

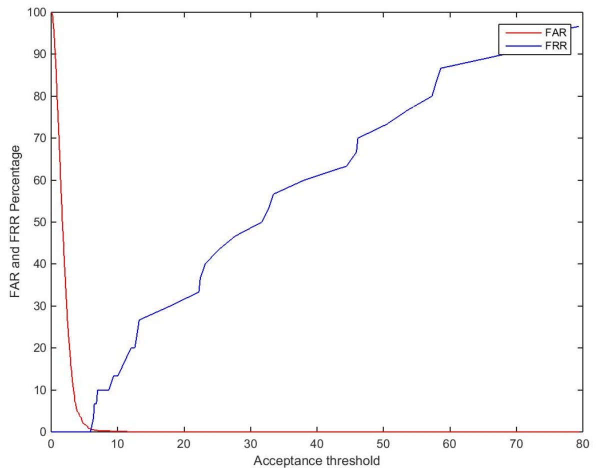

Fig. 3. Maximum scores picked as threshold (Equation 1)

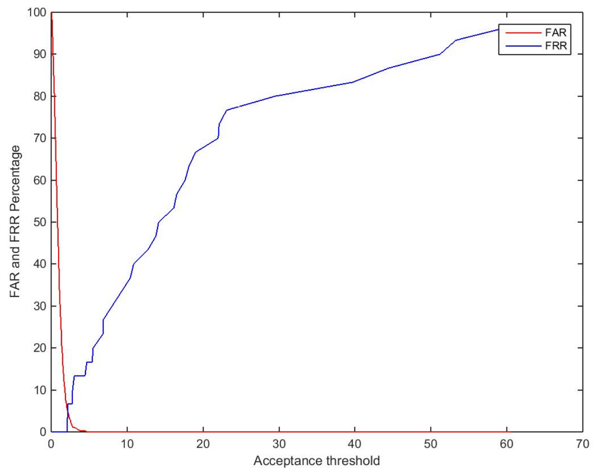

Fig. 4. Average scores picked as threshold (Equation 1)

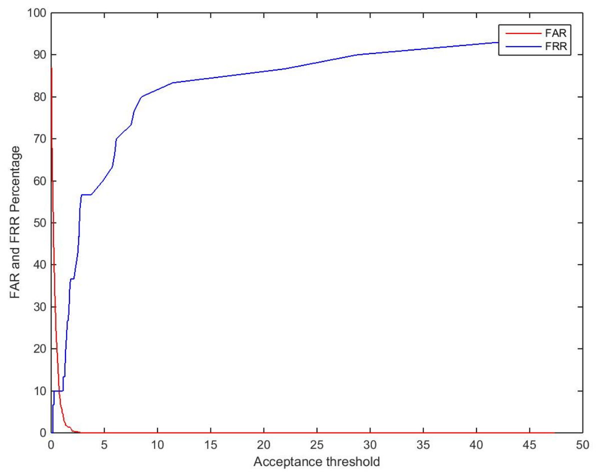

Fig. 5. Minimum scores picked as threshold (Equation 1)

1) Threat Model: The attacker's main aim is two fold: (i) to impersonate a genuine user, and (ii) to learn the key between the server and any victim user for eavesdropping purposes. We do not assume a secure channel. Thus, the attacker can obtain all protocol messages including the hash values and HMACs exchanged between the user and the server. Consequently, the attacker learns the number of minutiae used for the key agreement. The attacker may apply brute-force attack in passive mode by making use of the exchanged messages and try to guess the key. Similarly, in order to impersonate a genuine user, the attacker may apply a replay attack in active mode. However, our protocol resists these type of attacks to some extend as discussed in the upcoming subsections.

2) Resistance Against Brute-Force Attacks: The attacker can always launch a brute-force attack by trying all possible key combinations. Since the key is 256 bits long, this attack is infeasible. However, in this section, we will give a more intelligent brute-force attack by making use of the protocol messages.

This attack is applied by generating all possible minutiae locations and types. One fingerprint can have at most 512 x and y values, because of the sizes of the fingerprints in our database. A minutia can have two different types: *end* or *bifurcation*. It means that the attacker must generate $512 \times 512 \times 2 = 2^{19}$ points and hash them once and twice. Due to the fact that the user sends the genuine and fake minutiae list to the server, attacker's search space decreases from 2^{19} to $|Q_u|$. However, our analysis shows that this brute-force attack is still infeasible as discussed below.

The attacker has the set of genuine and fake minutiae points sent by the user, Q_u, and the number of minutiae, with which the key is generated, n_{com}^{key}. In order to find the generated key, the attacker should try all possible subsets of the set Q_u with size n_{com}^{key}, yielding the attack complexity att_c (Equation 3). For instance, if the user sends a list of 440 points, i.e. $|Q_u| = 440$, in which 40 of them are genuine, i.e. $|G_u| = 40$, and if the key agreement is completed with 16 common minutiae, i.e. $n_{com}^{key} = 16$, then the attack complexity becomes $\binom{440}{16} = \frac{440!}{16! \times (440-16)!} \cong 2^{96}$.

$$att_c = \binom{|Q_u|}{n_{com}^{key}} = \frac{|Q_u|!}{n_{com}^{key}! \times (|Q_u| - n_{com}^{key})!} \qquad (3)$$

In order to calculate the overall attack complexity of the system, the combination in Equation 3 is calculated after each key agreement. Then, we take the average of the complexity results. The analysis shows that the average attack complexity of the system is 94 bits (i.e. it requires 2^{94} hash and HMAC verifications) on the average. As discussed in [20], even with custom hardware implementation, computation of one block of HMAC-SHA256 takes approximately 0.8977 microseconds. Thus, the abovementioned complexity corresponds to 5.6×10^{14} years of attack. As a result, we can conclude that our protocol efficiently resists intelligent brute-force attacks.

3) Resistance Against Replay Attack and Impersonation: The aim of replay attack is to impersonate a genuine user and get the legitimate key. In order to do this, the attacker replays the previously exchanged messages between the victim user and the server. The attacker needs to know the genuine minutiae points to effectively calculate the generated key; otherwise, (s)he must try all possible combinations out of Q_u. Since the attacker does not know the genuine minutiae points, the complexity of this attack becomes the same as that of the brute-force attack given in Equation 3.

Moreover, the attacker may use his/her own fingerprint instead of the genuine user's fingerprint. The resistance of the protocol against this type of classical impersonation attack is shown to be very low since the FAR is 0.57%. The readers should also note that such counterfeiting attacks are general problems of all biometrics and related protocols; not specific to our one.

4) Randomness of the Generated Keys: The entropy measures the randomness of the keys. In this analysis, Shannon's Entropy values are calculated using Equation 4, in which K_i represents the i^{th} bit of the key.

$$H = -\sum_i P(K_i) log_2 P(K_i) \qquad (4)$$

As in the sample set, we use 300 keys (30 subjects, 10 keys per subject). The entropy values of these keys are given in Figure 6. The more the entropy value approaches to 1, the more random the key is. As can be seen in this figure, 83.67% of the keys have entropy values that are greater than 0.994, and also all of the keys have entropy values that are greater than 0.98, which implies very good randomness. It is important to note that these keys are generated by hashing the common minutiae. It can be normal to have high entropy for the hash results, because the hash functions kind of randomize the input string. Therefore, the entropy values of the concatenation of common minutiae are calculated as well. The concatenation is as follows, $x||y||type$. The entropy values of these concatenations are given in Figure 7. Although the entropy values decrease a little, 92.3% of the keys have entropy value above 0.98. Thus, they are still random enough.

5) Distinctiveness of the Generated Keys: Due to the fact that the fingerprints are time invariant biometrics, it is important to have a different key in each agreement. In each attempt, generating the same key is undesirable, because compromise of a key should not risk the confidentiality of the messages in other sessions. The minutiae quality and ordering change according to the fingerprint scanner, pressure of the finger on the scanner, acquisition environment, etc. This situation has both negative and positive effects on the key generation process. The negative effect is the difficulty of agreement on the same key in a protocol run. On the other hand, the positive effect is the generation of different keys in each attempt. In order to measure the difference of the keys for the same user, we calculate the Hamming distances of the keys of the same user after different protocol runs. As can be seen in Figure 8,

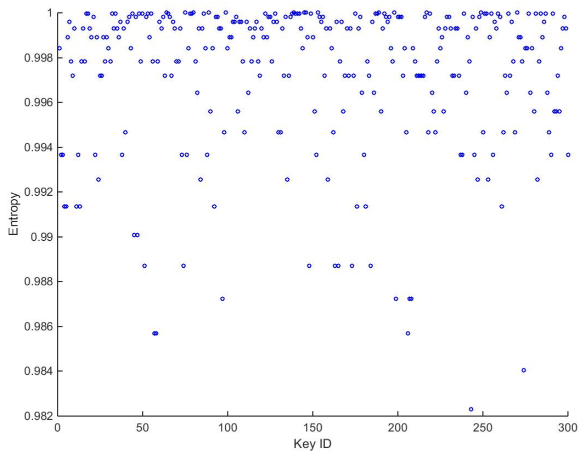

Fig. 6. Entropy values of the generated keys

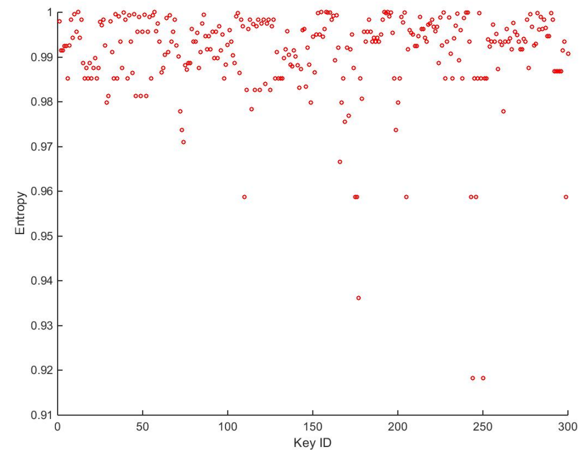

Fig. 7. Entropy values of the minutiae concatenations

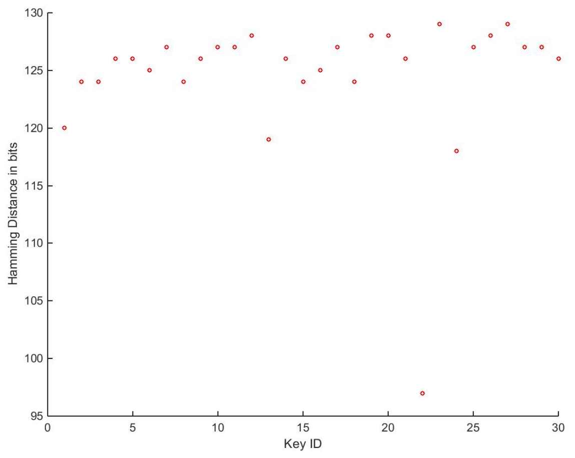

Fig. 8. Average Hamming distances of the same users' keys

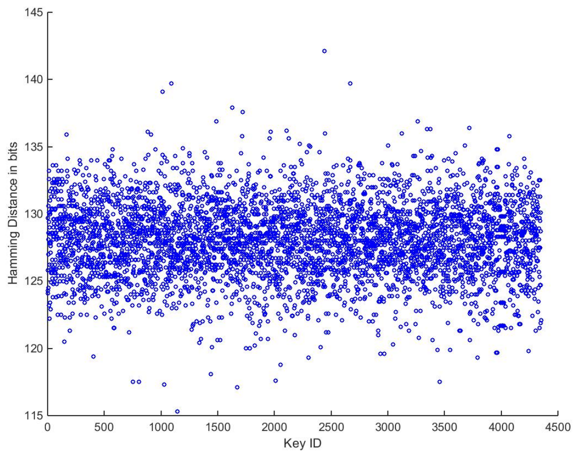

Fig. 9. Average Hamming distances of the different users' keys

the average Hamming distances of the keys vary between approximately 120-130 bits out of 256 bits for each user. Also the minimum difference is 97 bits. It means that the users have distinct keys after each key agreement phase. Additionally, as can be seen in Figure 9, the average Hamming distances of the keys that are generated for different users vary between approximately 120-135 bits. These values are very close to the average Hamming distances of the same users' keys. It means that we cannot decide if any two keys belong to the same user or different users by looking at their similarities or differences.

V. Conclusion and Future Work

In this paper, we proposed a novel secure key agreement protocol using unordered feature sets of biometric traits. This protocol is exemplified using the fingerprint biometrics. The key is generated by making use of minutiae points in the fingerprint; no random data is used while generating the key. This way, the user is strictly bound to the cryptographic key. Moreover, there is no need to store any helper or random data other than the biometric template of the user at the server side.

Our system uses hash functions and threshold mechanisms while generating the keys. In addition, we carefully designed a fake minutiae generation strategy such that the fake minutiae hide the genuine minutiae without being confused with the genuine minutiae. For this purpose, we defined the concept of *neighborhood relation*. With the help of the neighborhood relation, the fake minutiae increase the verification performance of the system while not leaking any information to the attacker.

We analyzed the security performance of our protocol in different aspects. Our results showed verification performance of 0.57% EER. The resistance of our protocol against intelligent brute-force, replay and impersonation attacks is also analyzed. Such attacks require 2^{94} trials on the average, which is shown to provide good computational security. In addition, we employed entropy-based randomness analyses of the agreed keys. Our analyses showed that 83.67% of the keys' entropy values are above 0.994 and all of the keys' entropy values are above 0.98, which implies that the keys are random enough to be used as cryptographic keys. Besides, the distinctiveness of the generated keys is measured using the Hamming distance metric. Our Hamming distance-based analyses showed that the same users' and different users' keys are quite indistinguishable from each other.

As a future work, the template renewal process on the server side can be designed. In other words, templates can be cancelable when needed. Moreover, our protocol can be adopted to other biometrics with ordered set of features, such as the iris biometrics.

References

[1] U. Uludag, S. Pankanti, S. Prabhakar, and A. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–960, June 2004.

[2] D. Karaoğlan and A. Levi, "A survey on the development of security mechanisms for body area networks," *The Computer Journal*, vol. 57, no. 1, pp. 1484–1512, 2014.

[3] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *6th ACM Conference on Computer and Communications Security*. New York, NY, USA: ACM, 1999, pp. 28–36.

[4] S. Kanade, D. Camara, E. Krichen, D. Petrovska-Delacretaz, and B. Dorizzi, "Three factor scheme for biometric-based cryptographic key regeneration using iris," in *Biometrics Symposium*, 2008, pp. 59–64.

[5] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1081–1088, Sept 2006.

[6] A. Stoianov, "Security of error correcting code for biometric encryption," in *Eighth Annual International Conference on Privacy Security and Trust*, 2010, pp. 231–235.

[7] C. Rathgeb and A. Uhl, "Statistical attack against iris-biometric fuzzy commitment schemes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2011, pp. 23–30.

[8] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[9] K. Nandakumar, A. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744–757, 2007.

[10] U. Uludag and A. Jain, "Securing fingerprint template: Fuzzy vault with helper data," in *Computer Vision and Pattern Recognition Workshop*, June 2006, pp. 163–163.

[11] P. Mihailescu, "The fuzzy vault for fingerprints is vulnerable to brute force attack," *CoRR*, vol. abs/0708.2974, 2007.

[12] W. Scheirer and T. Boult, "Cracking fuzzy vaults and biometric encryption," in *Biometrics Symposium, 2007*, Sept 2007, pp. 1–6.

[13] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," pp. 68 190O–68 190O–7, 2008.

[14] C. Örencik, T. B. Pedersen, E. Savaş, and M. Keskinöz, "Improved fuzzy vault scheme for fingerprint verification," *International Conference on Security and Cryptography*, 2008.

[15] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC-2104, US, 1997.

[16] A. M. Bazen and S. H. Gerez, "Fingerprint matching by thin-plate spline modelling of elastic deformations," *Pattern Recognition*, vol. 36, no. 8, pp. 1859–1867, 2003.

[17] "Neurotechnology Verifinger sample db," http://www.neurotechnology.com/, accessed: 2015-05-01.

[18] "Cross Match Verifier 300 lc," http://www.crossmatch.com/verifier-300-lc/, accessed: 2015-05-01.

[19] National Institute of Standards and Technology, *FIPS PUB 180-1: Secure Hash Standard*, 1995.

[20] M. Juliato and C. Gebotys, "FPGA implementation of an HMAC processor based on the sha-2 family of hash functions," University of Waterloo, Tech. Rep., 2011.