Pattern Recognition 111 (2021) 107733

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Biometric key generation based on generated intervals and two-layer error correcting technique

Peiyi Wang[a], Lin You[a,*], Gengran Hu[a], Liqin Hu[a], Zhihua Jian[b], Chaoping Xing[c]

[a] School of Cyberspace Security, Hangzhou Dianzi University, Hangzhou, 310018, China
[b] School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, 310018, China
[c] School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China

ARTICLE INFO

Article history:
Received 22 March 2019
Revised 1 June 2020
Accepted 29 October 2020
Available online 30 October 2020

Keywords:
Feature distance
Generated interval
Bio-key
Biometric
Code

ABSTRACT

As for a biometric key, key management and biometric data security are both important. Existing bio-key generation methods are usually based on the biometric templates or features directly, it may expose user's biometric data and will further make the biometric data permanently unusable for his secure identification recognitions. In this paper, a fingerprint bio-key generation approach using the feature distance is proposed. We utilize the relative distances among user's fingerprint minutiae to generate a unique bio-key. Such bio-key is determinable and recoverable via the generation interval scheme. In addition, we use a two-layer error correcting technique to guarantee a better reliability during the data transmission. The experimental results positively show that our approach can ensure higher security of the bio-key and guarantee a good key regeneration rate. Besides, the storage of the original bio-key or any fingerprint template is unnecessary.

1. Introduction

With the continuous promotion and progress of science and technology, big data, cloud computing, IoT and blockchain have been changing the human society and demonstrate a good developing prospect and future. At the same time, information security and digital identity become particularly important with unprecedented opportunities and challenges in such an era. The most important identity information is our inherent biometric information which is unchangeable and very suitable for convenient identity authentication, but it may expose its intrinsic defects that makes the biometric features be easily leaked intentionally or unintentionally and used by criminals for illegal activities. An effective combination of biometric technology with cryptographical technology, called biometric encryption technology, can not only ensure the original advantages of both methods but can also make up for each other's shortcomings. The core issue of the biometric encryption technology is to keep the balance between the fuzziness of biometric features and the accuracy of cryptography.

Secret key is the most important element to a cryptographic algorithm, and it is usually more than one hundred bits long for security and should be securely stored in a hardware storage medium, which is not only inconvenient for use but is also easy to be leaked once the storage medium is stolen or lost. To deal with this issue, a lot of work has been done on generation of secret keys from biometrics. The measure that using biometrics to generate secret keys mainly concerns two key points. Firstly, we should ensure the accuracy of the biometric secret key (abbreviated as "bio-key") and protect user's biometric data from being destroyed, stolen or lost at the same time, or the leakage of the related data will not result in the invalidation of the biometrics. Secondly, in order to reduce the influence of the extracted biometric features at other times on the generated secret keys, the generation method should also be fault-tolerant so that it can reduce the intra-variations of the extracted biometric features which may be caused by some environmental or man-made factors.

The main contribution of this paper is the proposal of a new approach for the generation of bio-keys from human fingerprints. We use the feature distances between fingerprint minutiae to calculate the key bit-string and achieve template-free by combining it with the generated interval scheme.

The remainder of this paper is as follows: In Section 2, we introduce a lot of recent work related to biometrics-based cryptographic key generation. Section 3 addresses our proposed approach, and the whole implementation procedures will also be described in detail in this section. Practical detail experiments on our approach are carried on Section 4. Section 5 gives the evaluation

* Corresponding author.
E-mail addresses: wangpywm@gmail.com (P. Wang), mryoulin@gmail.com (L. You), grhu@hdu.edu.cn (G. Hu), huliqin@hdu.edu.cn (L. Hu), jianzh@hdu.edu.cn (Z. Jian), xingcp@ntu.edu.sg (C. Xing).

https://doi.org/10.1016/j.patcog.2020.107733
0031-3203/

and security analysis on our approach. In addition, we will give a comparison of our approach with other recent valuable work in some key aspects including false acceptance rate (FAR), false rejection rate (FRR) and (bio-)key regeneration rate (KRGR), the renewability (R-New), computational cost and size. Four application scenarios are described in Section 6. Finally, the conclusion remarks come in Section 7.

2. Related work

A lot of methods have been put forward for biometric encryption or bio-key generation. In 1996, Tomko et al. proposed the concept of bio-encryption for the first time and gave a key generation system [1]combined with fingerprint features directly. In 2009, a statistical quantification mechanism was adopted by Ong et al. [2] to reduce the intra-variation of human dynamic signature in order to generate a more reliable signature template. Using the interval quantization based on a so-called method "context-based texture analysis" [3], Rathgeb et al. put forward a key generation scheme based on iris [4], and they eventually achieved the revocability of the bio-key by encoding the iris features. However, their method was based on the position information of the selected stable iris features while it may expose the iris data indirectly. In 2004, Dodis et al. defined two vital models [5]: the fuzzy extractor and the security sketch, which could ensure some fault tolerance and reliability respectively. The fuzzy extractor is basically equivalent to a probability function model: for an input biometric value w, a uniform random number series R is extracted which doesn't need to be stored, and later this R can be regenerated by an input value w' if it is close enough to w. The authors pointed out that the extracted R could be used as a key in the sense of cryptography, but they did not describe in detail how such R was constructed from the input biometric features. In 2008, Zhang et al. proposed an iris authentication scheme based on fuzzy extractor [6]. Using iris templates, Mariño et al. put forward a fuzzy extractor-based crypto-biometric scheme [7]. Mariño's scheme can be used by a user to retrieve a secret or a previously saved key by using his own iris template. In fact, Mariño's scheme is just like a fuzzy vault based on iris template.

For sake of biometric template security, much work has been made on the template-free schemes and some proposed approaches make the storage of templates or original generated bio-keys become unnecessary steps. Sheng et al. [8] proposed a reliable key generation method using fingerprint orientation fields and it did not require to store biometric templates, while it may not only lead some high FAR but also may not produce long cryptographic keys since the orientation fields only represent some small part of fingerprint information. In 2014, Wang et al.[9] constructed a cancelable fingerprint template based on a curtailed circular convolution and the alignment-free of the fingerprint was achieved. In 2015, Leng et al.[10] combined the 2DPalmHash code, fuzzy vault and the cancelable mechanism to construct the palmprint template and then proposed a hybrid cancelable palmprint-coded fuzzy vault. Sheng et al. [11] presented an encryption key generation method based on the statistical features of biometric data by developing a semisupervised data clustering scheme to model the user variations on both single biometric features and feature subsets, and they experimentally showed that their method could achieve good performance with some signature databases. They claimed that it could deliver a key with good consistency, discriminatory and entropy. However, they didn't give a detail description or a practical example about how their key was generated.

In [12], Partheeba et al. utilized the SIFT algorithm to extract fingerprint minutiae and applied the VLSB steganography to hide the minutiae template for the generation of a shared fingerprint bio-key by two users. But their method required that the extracted fingerprint image have high quality and they did not give a practical bio-key generation description or realizable algorithm. In 2017, Lavinia et al. [13] proposed a fuzzy extractor-based bioPKI skeleton in which its private key was a biometric key generated from the fingerprint template, iris template or the fused template of both fingerprint and iris, and they also provided the detailed experiments and analysis of the possible spoofing scenarios for the security of their bioPKI. But the authors did not give a description for their biometric key generation in a detailed algorithmic or mathematical form. Using an interval optimized mapping bit allocation method, Karimian et al. proposed a novel key generation approach called IOMBA which can produce a key from user's electrocardiogram (ECG) [14]. As they described, we know that ECG still has the challenges with respect to immunity to noise, abnormalities and etc. One year later, Moosavi et al [15] also proposed a cryptographic key generation approach based on several ECG features and experimentally showed that their approach had better P-value NIST pass rates and was approximately 1.8 times faster than the existing singleton ECG-based key generation approaches, but they did not test their KRGR. In addition, the acquisition of ECG is more complicated, inconvenient and time-consuming than the other biometrics such as fingerprint, finger vein and iris. In [16,17], Verma et al. constructed a biometric key generation scheme from fingerprint hologram and it can be used for optical image encryption. They theoretically showed that the fingerprint biometric keys were unique with high discrimination ability for each person and the keys' matching probability between two different persons are minimal. They simulated their approaches' effectiveness about their fingerprint key generation method, but they did not experimentally test the fingerprint KRGRs.

In order to make up for the fuzzy extractor's defect that the unique helper data of users is essential to it, Seo et al.[18] presented a biometric-based key derivation function (BB-KDF) which made the users be able to derive their cryptographic keys solely from personal biometric data without any helper information. The key part of the BB-KDF is the round function with a preset threshold vector for any input biometric vector. But this round function may induce an imposter to succeed in deriving a correct key while a genuine user may not derive his correct key. In addition, to successfully regenerate a correct key, it must be assumed that as many minutiae as before could be exacted at the stage of re-deriving the key. In 2018, Wu et al. [19] gave a new bio-key generation scheme named FVHS based on finger vein features. The authors combined machine learning, biometrics and cryptography technologies together to extract a special feature vector from user's finger vein and then processed it to generate a bio-key with a fixed length, but the stability of human's finger vein which guarantee the appearance of the extracted features in their work remains questionable. In this year, Anees et al. [20] developed a unified framework for generating cryptographic keys from facial features, and they simulated their approach's recognition rate and showed the generated key had good random and robustness. In addition, they experimented their ELBP's recognition rate which may basically represent the key generation rate. One year later, based on the straight line attributes (line lengths and their angles) between minutia points, Panchal et al. [21] proposed a novel approach for generating a threshold-free cryptographic biometric key (called bio-crypto key) by merging all bits in the three feature sets of the straight line attributes from three minutia types, respectively, and they also used Reed-Solomon encoding to generate a codeword for encryption and especially for the retrieval of user's bio-crypto key. The authors claimed that their approach did not require to store user's biometric template or the generated key. But, we have noticed that the three feature sets may be the same set and their approach may have the risk of disclosing the fingerprint features's straight line attribute information. An attacker can first apply user's ciphertext to

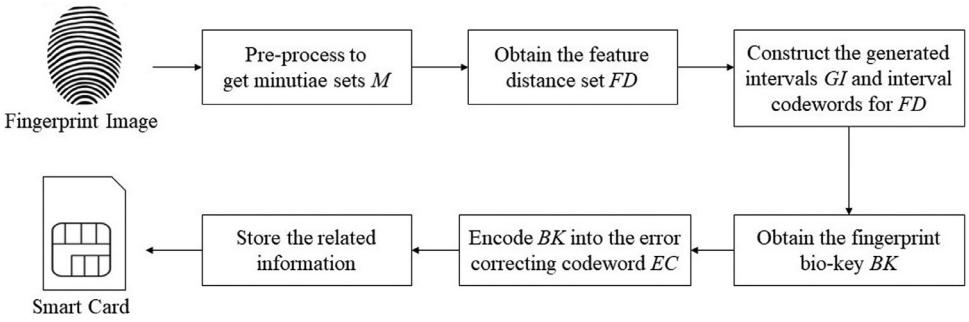

Fig. 1. The process of our bio-key generation approach.

separate the codeword from user's final ciphertext, then obtain the line attributes from the codeword using Reed-Solomon decoding, and finally he may successfully reconstruct user's bio-crypto key from the line attributes if he can obtain the parameters for substitution or expansion. In addition, the approach had a considerable burden of keeping a S-box, two random arrays and the parameters of the encoding scheme secretely.

In 2019, by utilizing the convolution coding principle, Panchal et al. [22] further proposed a biometric codeword generation method which can generate a unique codeword, and the hash value of this codeword together with some private parameters can be used as a cryptographic key. They experimented out that the genuine rate was up to 95.12% while the false rate was 0%. This method eliminated the storage of any biometric template or secure key and it was effective enough to be applied in any security system. But there existed some issues such as some biometric images may have no center points, and one may not extract out the same biometric minutiae set as the previous gotten minutia set when he is to regenerate the bio-key.

As we have described above, we know that a lot of valuable work has been done on biometric-based key generations, however, some of them were found to be lack of more practical simulations for the bio-key generation. In Sections 3 and 4, we will propose a novel biometric-based key generation method and some practical simulations will be made on our approach with fingerprints.

3. Proposed approach

At the bio-key generation stage, the user's fingerprint minutiae are firstly extracted, then, we calculate the feature distances between some two minutiae. In the generated interval scheme, each group of the feature distances constructs a unique generated interval with its cancelable interval codewords. For a better accuracy, we choose a two-layer error correcting technique to correct the mistakes appeared in data transmission. Finally, user's related information is stored in the smart card for the bio-key regeneration. In the last part of this section, we will describe the whole procedure of the bio-key regeneration. Fig. 1 shows the flow of our bio-key generation approach.

3.1. Fingerprint minutiae

A human fingerprint is normally comprised of ridges and valleys [23]. The ridges are the dark lines running through the fingerprint image while the valleys are the light lines adjoined between ridges. As is well-known, human fingerprints vary from each other and keep unchanged during human whole life. The integral topology of fingerprint ridges and valleys together with general direction of lines are called the global features. Correspondingly, the local features include minutiae like core, delta, ending and bifurcation, and the features also include distributions like gradient, orientation, frequency and etc.

3.1.1. Preprocessing

The fingerprint image extracted from the user should be preprocessed through the eight steps as shown in the following Fig. 2.

3.1.2. Minutiae set

After preprocessing, we extract all the ending, bifurcation, core and delta points from the fingerprint binary image and record their information, then we can get the fingerprint minutiae set $M = \{\alpha_i = (x_i, y_i, t_i, \theta_i) | i = 1, 2, \ldots, N\}$, where N is the total minutia number, x_i is X-coordinate, y_i is Y-coordinate, $t_i = 0, 1, 2, 3$ is the minutia type representing ending, bifurcation, core or delta point, respectively, and $\theta_i \in [0, 2\pi]$ denotes the orientation field value. Fig. 3 shows an example of the original fingerprint image and its corresponding pre-processed images with different types of minutiae are marked.

3.2. Minutiae to feature distance

During the acquisition and alignment processes, fingerprint minutiae will be faced with non-negligible changes caused by image rotation and transformation to some extent [24], thus, we choose the feature level rotation rather than the image level rotation and utilize the feature distance to make the further comparison. In addition, we will store the differences of the minutia types and the differences of the orientation fields as the auxiliary data so that we can regenerate the bio-key more accurately.

3.2.1. Minutiae statistics

We first extract minutiae respectively from the user's m training image samples of the same fingerprint and get m minutiae sets $M_k = \{\alpha_{ki} | i = 1, 2, \ldots, N_k\}$, where $\alpha_{ki} = (x_{ki}, y_{ki}, t_{ki}, \theta_{ki}), k = 1, 2, \ldots, m$. Then, make a statistics about the occurrence times of the same minutia in different sets and arrange the minutiae from the highest frequency to the lowest frequency.

We choose the first $(n+1)$ minutiae with the highest frequency for each minutiae set M_k, then randomly select one minutia that appeared in each training sample as the top minutia α_{ki_0} and if there is no such minutia, the one with the highest frequency will be selected. The m sets of the selected minutiae are denoted as $SM_k = \{\alpha_{ki_0}, \alpha_{ki_1}, \ldots, \alpha_{ki_n} | \{i_0, i_1, \ldots, i_n\} \subseteq \{1, 2, \ldots, N_k\}\}, k = 1, 2, \ldots, m$. Notice that if one minutia is not appeared in the current minutiae set, we will set its corresponding selected minutia to be *None*. The value of n determines the bio-key size and it is no more than 25 in our experiments.

3.2.2. Feature distance set

Now we calculate the Euclidean distances between the top minutia $\alpha_{ki_0} = (x_{ki_0}, y_{ki_0}, t_{ki_0}, \theta_{ki_0})$ and all the other minutiae $\alpha_{kj} = (x_{kj}, y_{kj}, t_{kj}, \theta_{kj})$, $j = 1, 2, \ldots, n$ $(j \neq i_0)$ in SM_k as in Eq. (1). For the sake of a better key regeneration, the differences between the minutia types and the difference between the orientation fields are

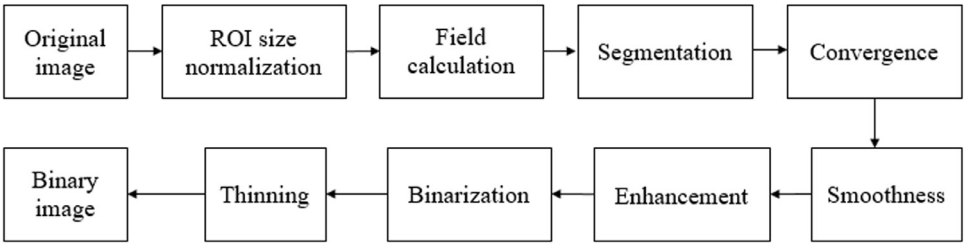

Fig. 2. Fingerprint preprocessing.

Fig. 3. Original image, preprocessed result and marked minutiae.

also calculated correspondingly as in Eq. (2), serving as the auxiliary data. Note that in order to standardize all the feature distances to adapt the following steps, we apply a modular operation to the Euclidean distances. The modulus is the feature space size 2^p which is determined by the code length p. Note that p is a positive integer which is usually no more than 16, and it is also used in Section 3.3.2. In our experiments, p is set to be 8 bit and so that the modulus is 256.

$$d_{kj} = \sqrt{(x_{kj} - x_{ki_0})^2 + (y_{kj} - y_{ki_0})^2} \ mod\ 2^p \tag{1}$$

$$\triangle t_{kj} = t_{kj} - t_{ki_0}, \quad \triangle\theta_{kj} = \theta_{kj} - \theta_{ki_0} \tag{2}$$

Now, we define that a feature distance with three parameters: Euclidean distance, the difference between the minutia types and the difference between the orientation fields. Finally, we can obtain the user's $m$feature distance sets written as $FD_k = \{(d_{kj}, \triangle t_{kj}, \triangle\theta_{kj}) | j = 1, 2, \ldots, n\}, k = 1, 2, \ldots, m$. Notice that if one minutia is not extracted in the current minutiae set, we will set its corresponding feature distance with the top minutia to be *None*.

3.3. Generated interval scheme

In this section, we give an example to explain the generated interval scheme.

We assume that there is one feature object X with its l times' extraction values being $\{x_1, x_2, \ldots, x_l\}$. Let $x_{\min} = \min(x_1, x_2, \ldots, x_l), x_{\max} = \max(x_1, x_2, \ldots, x_l)$ and μ be a predefined quantization value for X, then a unique interval is constructed for X as $[x_{\min} - \mu, x_{\max} + \mu]$ which we call the generated interval of X. After that, the generated interval is encoded to obtain its unique codeword which is selected at random. Note that if the number of the generated intervals is more than one, then their corresponding codewords should be all isometric.

For example, the values of X are $\{63, 61, 61, 61, 62, 60, 61, 62\}$ with $l = 8$ and the quantization value is predefined as $\mu = 2$. Apparently, $x_{\max} = 63, x_{\min} = 60$, and the generated interval for X is to be $[58, 65]$. We can select "01" as the codeword for this generated interval.

3.3.1. Generating intervals

Based on the above conception, we construct the generated intervals for the feature distance sets $FD_k = \{(d_{kj}, \triangle t_{kj}, \triangle\theta_{kj}) | j = 1, 2, \ldots, n\}, k = 1, 2, \ldots, m$. Let $d_{j,\min} = \min(d_{1j}, d_{2j}, \ldots, d_{mj})$ and $d_{j,\max} = \max(d_{1j}, d_{2j}, \ldots, d_{mj})$, then for each j, the generated interval for the set $(d_{1j}, d_{2j}, \ldots, d_{mj})$ is $[L_j, R_j]$ with L_j and R_j defined as the following Eq. (3):

$$\begin{cases} L_j = d_{j,\min} - \delta \\ R_j = d_{j,\max} + \delta \end{cases} \tag{3}$$

Where $d_{j,\min} = \min(d_{1j}, d_{2j}, \ldots, d_{mj}), d_{j,\max} = \max(d_{1j}, d_{2j}, \ldots, d_{mj})$,and δ is the quantization parameter which is pre-tested and used to extend the interval. The value of δ should be tested out by several experimental tests in order to achieve the fault tolerance and it is usually less than the one-tenth of 2^p. Note that, all the differences of the orientation fields in the auxiliary data are also replaced by their corresponding generated intervals with no encoding.

For a better bio-key regeneration, we should arrange the generated intervals in order. Here the generated intervals are ordered based on the rules: 1) Arranging the intervals in the order of the values of the left boundary L_j from the smallest to the largest. 2) Arranging the intervals in the order of the values of the right boundary R_j from the smallest to the largest if their left boundaries are equal. Finally, the feature distances are transformed and arranged into a set of generated intervals denoted as in Eq. (4).

$$GI = \{[L_j, R_j] | j = 1, 2, \ldots, n\} \tag{4}$$

3.3.2. Interval encoding

Since fingerprint is almost never changed for the human whole life, once user's original fingerprint data is compromised, the fingerprint may become permanently unusable for his digital identity-related applications. The main point of a cancellable bio-key is that it can be stored and used in a kind of transformed form so that once the transformed bio-key is compromised, we can update the bio-key by changing the transform function or some related parameters.

In our approach, the generated interval $[L_j, R_j]$ $(j = 1, 2, \ldots, n)$ is encoded into a $p-$bit codeword bt_j which is chosen at random but isometric so that we can achieve the cancellability of the bio-

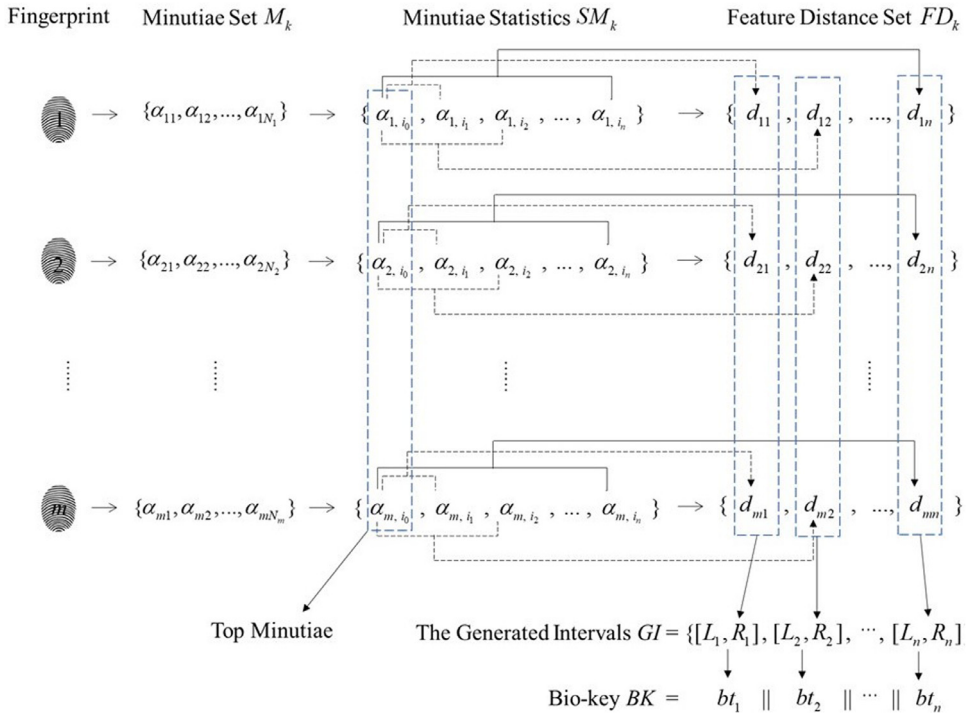

Fig. 4. Transformation diagram from the fingerprint to its corresponding bio-key.

key. Furthermore, all the codewords should be chosen to be different even if the generated intervals are similar or identical in value because our approach needs a certain number of successful mappings in the bio-key regeneration procedure which is described in detail in the following Section 3.6.

The original bio-key, denoted as *BK* in Eq. (5), is the concatenation of all the codewords of *GI* and its length is $p \times n$ bits. Besides, once the user's bio-key is lost, exposed or in danger, we can employ new codewords to update it immediately. In our experiment, we select $p = 8$ and transform user's fingerprint into an $8n$-bit bio-key in result.

$$BK = bt_1||bt_2||\cdots||bt_n \tag{5}$$

Fig. 4 is the diagram of the whole transformation from the fingerprint to the original bio-key *BK*.

3.4. Two-layer error correcting technique

In our proposed approach, we use a two-layer error correcting technique which combines Hadamard code with Reed-Solomon code to detect and correct unexpected errors that occur during the transmission for our fingerprint bio-key *BK*.

3.4.1. Two error correcting codes

In this section, the two error correcting codes applied in our approach will be separately introduced with examples.

Reed-Solomon code (RS code) is a BCH code and can be denoted as (r, s) with $r = 2^p - 1, s = 2^p - 1 - 2t$. The parameters represent that when we input s symbols, the output RS code will be r symbols and it can correct at most t errors with $2t$ verifying symbols, in which every symbol is a decimal number containing p bits information. The RS encoding is based on the polynomial division on the Galois field $GF(2^p)$ and we will take an example to illustrate its mechanism.

Assume that "2, 5, 4, 1, 6" is the input symbols while each symbol contains no more than 3 bits, then $p = 3, r = 7, s = 3$ and $t = 2$ are suitable for RS encoding. Let α be a root of the primitive polynomial $f(x) = x^3 + x + 1$in $GF(2^3)$, then $\alpha^3 + \alpha + 1 = 0$ and we get the generator polynomial $g(x) = \alpha^0 x^2 + \alpha^4 x + \alpha^3$ as in Eq. (6). Then, we transform the decimal symbols "2, 5, 4, 1, 6" into the binary bits "010, 101, 100, 001, 110" and map them on $GF(2^3)$ with reference to Table 1 to obtain the input polynomial $s(x) = \alpha^1 x^4 + \alpha^6 x^3 + \alpha^2 x^2 + \alpha^0 x^1 + \alpha^4$. According to the polynomial division on the Galois field, calculate the remainder as in Eq. (7) and we can obtain the remainder $s_r(x) = \alpha^3 x^1 + \alpha^1$. Finally, the polynomial $r(x) = \alpha^1 x^6 + \alpha^6 x^5 + \alpha^2 x^4 + \alpha^0 x^3 + \alpha^4 x^2 + \alpha^3 x^1 + \alpha^1$ is comprised of $s(x) \cdot x^{2t}$ and $s_r(x)$ as in Eq. (8) and the decimal symbols "2, 5, 4, 1, 6, 3, 2" of the coefficients of $r(x)$ is regarded as the RS code.

Table 1
Mapping on the Galois field $GF(2^3)$.

Coefficient	Relationship	Binary bit	Decimal symbol
0	0	000	0
α^0	1	001	1
α^1	α	010	2
α^2	α^2	100	4
α^3	$\alpha + 1$	011	3
α^4	$\alpha^2 + \alpha$	110	6
α^5	$\alpha^3 + \alpha^2$	111	7
α^6	$\alpha^4 + \alpha^3$	101	5

$$g(x) = (x - \alpha)(x - \alpha^2)\cdots(x - \alpha^{2t}) \tag{6}$$

$$s_r(x) = s(x) \cdot x^{2t} \ mod \ g(x) \tag{7}$$

$$r(x) = s(x) \cdot x^{2t} + s_r(x) \tag{8}$$

In the RS decoding and error correction process, the primitive polynomial $f(x)$ on $GF(2^3)$ with its primitive root α and the RS code are known in advance. The RS code are transformed into a polynomial $R(x)$ and the corresponding syndromes for $R(x)$ are calculated as in Eq. (9). If all the syndromes are 0, it indicates that there is no error in the RS code and we can obtain the original

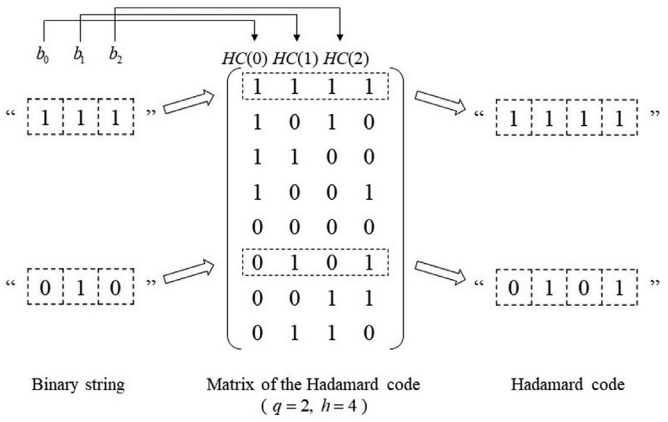

Fig. 5. Examples of Hadamard encoding process.

symbols directly form the first s symbols of the RS code. Otherwise, all the errors will be detected and corrected using the Berlekamp-Massey decoding algorithm [25].

$$s_i = R(\alpha^i), i = 1, 2, \ldots, 2t. \tag{9}$$

As for the example above, the RS code is transformed into the polynomial $R(x)$ and we assume that $R(x) = m_4x^6 + m_3x^5 + m_2x^4 + m_1x^3 + m_0x^2 + q_1x^1 + q_0$. The syndromes s_0,s_1 as in Eq. (10) are calculated according to Eq. (9). When the RS code is "2, 5, 4, 1, 6, 3, 2", the syndromes s_0,s_1 are both 0 and the decoded symbols is "2, 5, 4, 1, 6".

$$\begin{cases} s_0 = m_4\alpha^6 + m_3\alpha^5 + m_2\alpha^4 + m_1\alpha^3 + m_0\alpha^2 + q_1\alpha^1 + q_0 \\ s_1 = m_4\alpha^{12} + m_3\alpha^{10} + m_2\alpha^8 + m_1\alpha^6 + m_0\alpha^4 + q_1\alpha^2 + q_0 \end{cases} \tag{10}$$

When the RS code is "2, 6, 4, 1, 6, 3, 2", the corresponding syndromes are $s_0 = \alpha^6$ and $s_1 = \alpha^1$ so that the error is occurred at m_3 in the syndrome s_0 with the factor $s_0/s_1 = \alpha^5$. After the error detection, we can correct the symbol m_3 by substituting the right symbols in the RS code into Eq. (11) to obtain that $m_3 = \alpha^6$. Therefore, the RS code after the error correcting is "2, 5, 4, 1, 6, 3, 2" and the decoded symbols is also "2, 5, 4, 1, 6".

$$\alpha^6m_4 + \alpha^5m_3 + \alpha^4m_2 + \alpha^3m_1 + \alpha^2m_0 + \alpha^1q_1 + q_0 = 0 \tag{11}$$

Hadamard code is generated by Hadamard matrix $H_h(h = 2^q)$ which is comprised of "1" and "-1". However, the symbol "1" is mapped into "1" and "-1" is mapped into "0" in the practical applications. Hence, a pairwise-orthogonal h-order Hadamard matrix can generate $2h$ completely different codewords of h bit long. The minimum Hamming distance between any two different Hadamard codes is 2^{q-1} bit, hence, it can detect at most $2^{q-1} - 1$ bit errors, and correct $2^{q-2} - 1$ bit errors.

The process of the Hadamard encoding from the binary string $B = \{b_i | i = 0, 1, \ldots, q\}$ to its Hadamard code $HC_B = [h_0, h_{20}, \ldots, h_{2^q-1}]$ is denoted as Eq. (12) in which $\{h_0 = b_0, h_{2^i} = b_{i+1} | i = 0, 1, \ldots, q-1\}$ are the information bit in HC_B and the remaining bits are the verifying bits.

$$HC_B = Encode(b_0, b_1, ., b_q) \tag{12}$$

For example, let $q = 2, h = 4$,"111" and "010" are two 3-bit binary strings. The 4-order Hadamard matrix and the whole encoding process are shown in Fig. 5. The bits in the binary string corresponds to the first, 2^0th, 2^1th bit in Hadamard code successively, thus the corresponding Hadamard codes for "111" and "010" are "1111" and "0101", respectively.

The main idea of the Hadamard decoding is to find out a corresponding row in the pairwise-orthogonal Hadamard matrix whose Hamming distance with the received Hadamard code is less than 2^{q-2}. If it exists, the original information can be successfully decoded by extracting the information bit from the corresponding row. Otherwise, the decoding process is failed.

3.4.2. Implementation

For the purpose of an effective combination, the number of bits contained in a symbol of the RS code must be equal to the input bit number of the Hadamard code, that is $p = q + 1$. Here we give the diagram of the two-layer error correcting encoding process of the bio-key as Fig. 6.

Firstly, as the original bio-key is $p \times n$-bit long, we should transform the binary string into n decimal symbols with each of which being a p bit binary number. Secondly, in order to fulfill the input bits, we insert $(s - n)$ zeros ahead of the $n$decimal symbols, and then encode them into an original RS code which contains r symbols. The first $(s - n)$ zeros of the original RS code are removed and every symbol is encoded into a h-bit Hadamard code, respectively. Finally, the binary codeword string which is connected by all the Hadamard codes is transformed into the symbol form where p pit corresponds to one symbol and it is also the final two-layer error correcting codeword EC for the bio-key.

For example, assume that the bio-key is "010101100001110" and $p = 3, n = 5$. Firstly, we divide and transform the bio-key into the symbol form "2, 5, 4, 1, 6" where 3 bits correspond to one symbol. Obviously, it is unnecessary to insert any zero since $s = n = 5$. Then, according to the example above, we know that the RS code for "2, 5, 4, 1, 6" is "2, 5, 4, 1, 6, 3, 2". Encode the RS code symbols into the Hadamard codes one by one so that we can obtain the binary codeword string "010110101001001111000011001 01". Finally, we transform this string into its symbol form "0, 5, 5, 2, 2, 3, 6, 1, 4, 5" by complementing some zeros in the front of the string. Hence, the final two-layer error correcting codeword for the bio-key is "0, 5, 5, 2, 2, 3, 6, 1, 4, 5".

After the two-layer error correcting operation is performed, the user's original bio-key BK becomes a $\lceil (r - s + n) \times 2^q/p \rceil$ symbol codeword EC.

3.5. Storage of user's information

In order to make sure the correctness of the decoded bio-key, we also apply a hash function on the original bio-key BK as used in the literature [26]. Hence, when the decoded bio-key is correct, its hash value must be equal to the hash value of BK. In our approach, $SHA256$ is selected and the hash value of the original bio-key BK is denoted as $HK = SHA256(BK)$.

The information for a bio-key generation scheme generally contains four parts, that is, the generated intervals GI with corresponding codewords, the auxiliary data, the error correcting codeword EC and the hash value HK of the original bio-key BK, all of which will be stored in user's smart card at the end of the bio-key generation procedure.

3.6. Bio-key regeneration

In this section, we will mainly discuss the inverse procedures of our approach. Assume that the user wants to regenerate his bio-key BK with his fingerprint and his smart card, the bio-key regeneration procedure can be described as the following six steps and the flow diagram is embodied in Fig. 7.

(1). Extract and obtain the minutia set $M' = \{\alpha'_i = (x'_i, y'_i, t'_i, \theta'_i) | i = 1, 2, \ldots, N\}$ from the fingerprint, and then calculate the feature distances between each minutia α'_i and all the other minutiae $\alpha'_j (j \neq i)$ to obtain its feature distance set $FD'_i = \{(d'_{ik}, \triangle t'_{ik}, \triangle\theta'_{ik}) | k = 1, 2, \ldots, N-1\}, i = 1, 2, \ldots, N$.

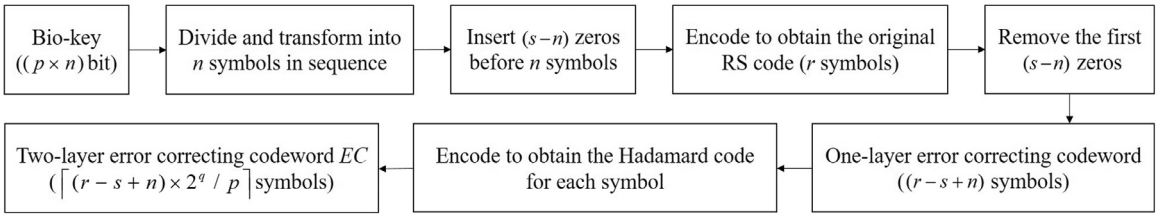

Fig. 6. Two-layer error correcting encoding process.

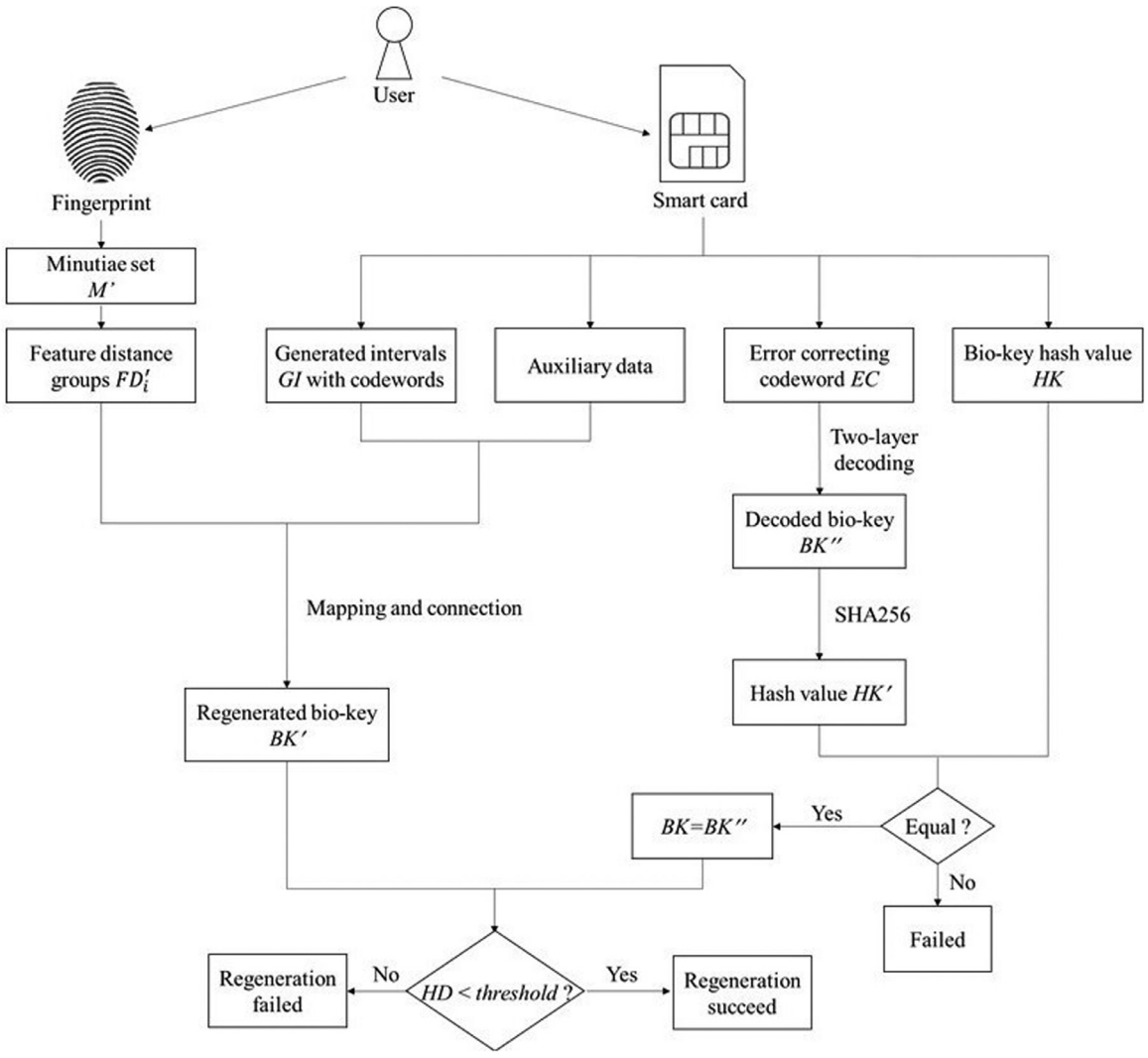

Fig. 7. Bio-key regeneration process.

(2). Read the user's information from his smart card. The information includes the generated intervals *GI* with the corresponding codewords, the auxiliary data, the error correcting codeword *EC* and the hash value *HK* of the original bio-key.

(3). A successful mapping means that the Euclidean distance of one feature distance locates in one generated interval, as well as the difference of the orientation field locates in its corresponding generated interval and the difference of the minutia type is equal to the type difference value in auxiliary data. Try to map the values in the feature distance set $FD'_i\,(i = 1, 2, \ldots, N)$ into the generated intervals *GI* and the auxiliary data, and record the successful mapping times for each group. The successful mapping is non-repeating.

(4). Select the feature distance set with the maximum successful mapping times and replace the mapped generated intervals with their corresponding codeword and the unmapped generated intervals with all zero codeword. The concatenation in order of all the codewords is regarded as the regenerated bio-key BK' of the user.

(5). Apply the two-layer decoding operations on the error correction codeword *EC* to obtain a decoded bio-key BK''. The hash value of $BK''$is denoted as HK'. If $HK' = HK$, then the decoding is successful and $BK = BK''$, otherwise, decoding is failed.

(6). If the above decoding is successful, we calculate the Hamming distance *HD* between $BK'$and *BK*. If *HD* is less than the predefined threshold, then the bio-key regeneration is successful, otherwise, the regeneration is failed.

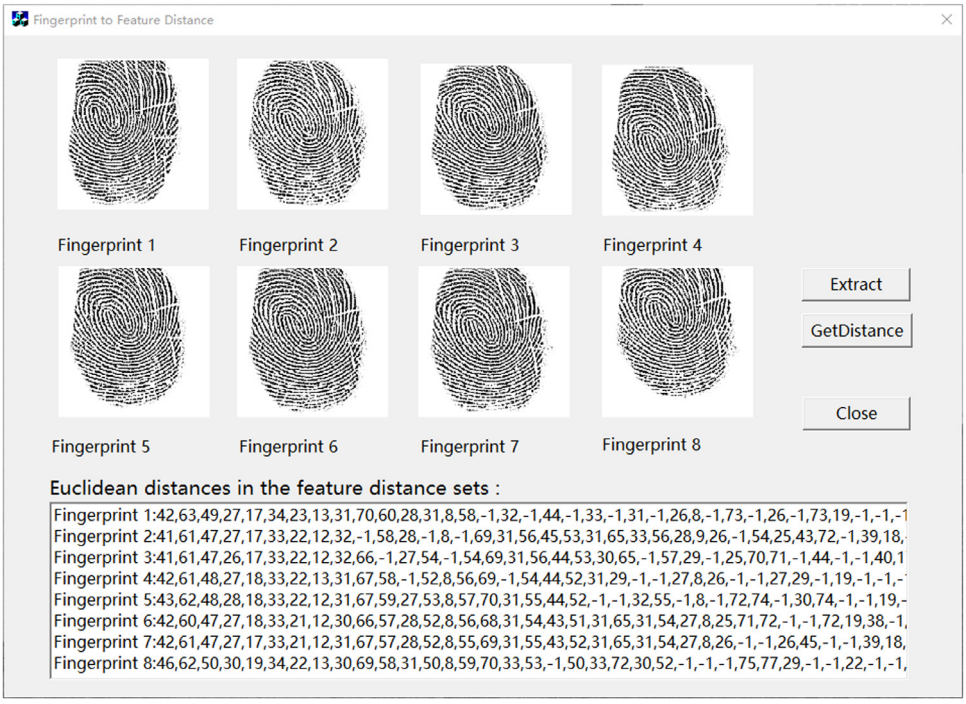

Fig. 8. Interface from fingerprint to feature distance.

4. Experiments

In this section, we will experiment with our approach in the computing environment with Windows10 64-bit, an Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz×4, and the used softwares including Visual Studio2015, MyEclipse 2016 Stable 1.0 and MatlabR2014a.

4.1. Key generation

4.1.1. Fingerprint to feature distance

Fig. 8 shows the interface of the extraction and computation procedures from the fingerprint images to the feature distance sets. Firstly, we extract 8 minutia sets from 8 fingerprint images of user's one finger, then click "GetDistance" button to obtain the feature distance sets and the corresponding Euclidean distances will be listed in the following blank.

4.1.2. Bio-key generation

The bio-key generation interface in our experiment is exhibited as Fig. 9. We should firstly input the feature distance sets obtained in the previous step as in Fig. 8 and the related Euclidean distances will be listed in the form. Next, click "Generate" button to generate the original bio-key binary string BK for the user and construct the generated intervals GI with corresponding codewords in the background. Then, we encode the bio-key BK into the two-layer error correcting codeword EC. Later, the system calculates the hash value HK of BK using the hash function SHA256 in the following step. Finally, the "Save" button is designed to store the user's related information as mentioned in Section 3.5 into the user's smart card. Our experiments show that it will take about 51.1 milliseconds (ms) to generate a 128 bit key from the input fingerprint image patterns.

4.2. Key regeneration

For the bio-key regeneration, we should input the user's fingerprint and read the information in user's smart card at the first step, then we calculate the feature distance sets and try to map the values in the sets into the generated intervals GI in the background. The feature distance set with the maximum successful mapping times will be selected and then the corresponding codewords will be concatenated to produce the regenerated bio-key BK' when we click the "ReGenerate" button. The "Decode" button will implement the two-layer error correcting decoding of the codeword EC to obtain a decoded bio-key $BK''$and the "SHA256" button produces the hash value of the decoded bio-key BK'' using SHA256 so that it will be verified whether BK'' is the original bio-key BK or not. If not, it will give a failed tip. Otherwise, click the "Check" button to calculate the Hamming distance between the original bio-key $BK(BK = BK'')$ and the regenerated bio-key BK'.

As we can see, the Hamming distance between BK and BK' is calculated as the determinant. If the Hamming distance is smaller than the predefined threshold, it will give a succeed tip and regenerate the user's bio-key BK successfully. Otherwise, the bio-key regeneration is failed. The whole interface of the bio-key regeneration is designed as Fig. 10 and the Hamming distance used in our example is 3 when the bio-key is 128 bits.

We noticed that, in our bio-key regeneration simulation, the resulted bio-key will be obtained in about 27.9 ms. It means that our bio-key generation costs 23.2 ms more time than our bio-key regeneration. This is mainly because that our approach will take much time to extract multiple training samples from the same input biometric feature during the bio-key generation, while it only needs to extract one sample from the input biometric feature during the bio-key regeneration.

5. Evaluation and analysis

We tested our proposed approach on FVC2004 DB3 and our fingerprint database HD-FP2015 DBv1 which has totally 600 fingerprint images from 30 different students, each of whom contributed 20 samples of his same finger. As for the performance evaluation, the bio-key regeneration rate and error correcting result will be demonstrated for the effectiveness. In addition, we will give a se-

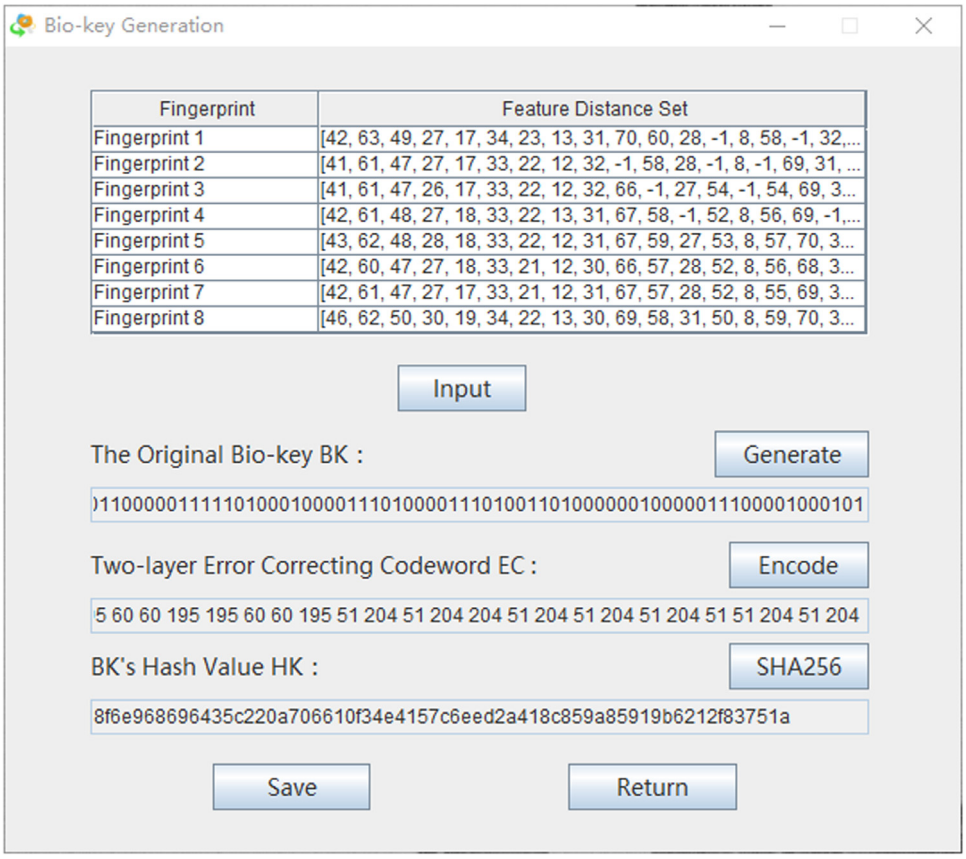

Fig. 9. Bio-key generation interface.

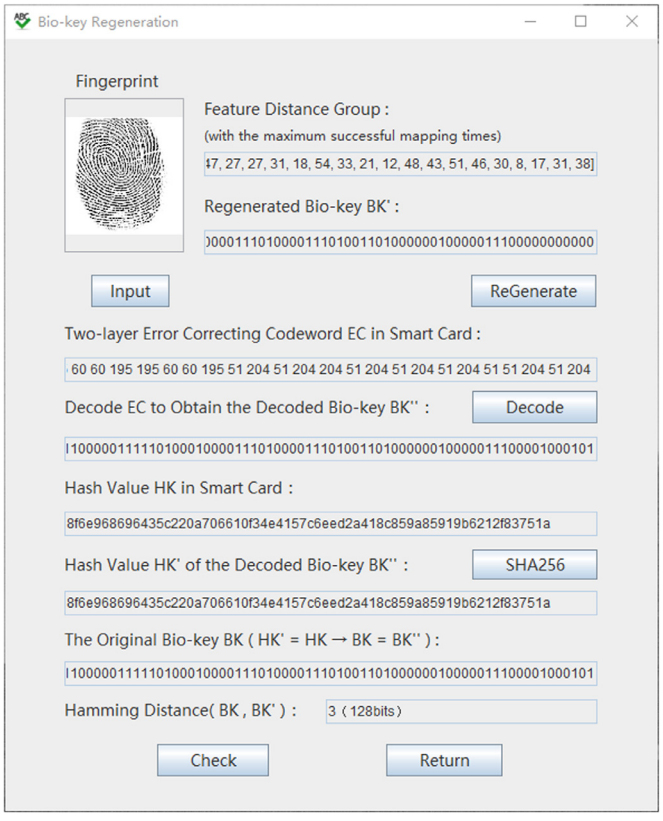

Fig. 10. The bio-key regeneration interface.

curity analysis on our approach. In our experiments, the parameters mentioned above are set as $m = 8$, $p = 8$ and $q = 7$. The predefined quantization parameter δ is tested from 0 to 20 in our experiments. Our experimental result is relatively better when $\delta = 5$.

5.1. Bio-key regeneration rate

The bio-key regeneration rate relates to the threshold values. In our approach, we choose the Hamming distance between the regenerated bio-key BK' and the original bio-key BK as a reference threshold. The bio-key regeneration rates which vary with the increase of Hamming distance for both genuine and imposter are plotted in Fig. 11. The bio-key is 128 bits with $n = 16$ in this section.

When the threshold is 15-bit long, the bio-key regeneration rate for the genuine user goes up to at least 92.92% while the bio-key regeneration rate keeps 2.58% for the imposter. In other words, the probability that the imposter regenerates the bio-key is only 2.58% while the probability for the genuine user is 92.92%. In addition, the bio-key regeneration rate for the fingerprint samples in FVC2004 DB3 is also at a high level as Fig. 11shows, the bio-key regeneration rate for FVC2004 DB3 is 91.14% when the threshold is 15-bit long.

5.2. Regeneration rates with increasing minutiae

In our approach, the size of the bio-key increases with the number n of the selected minutiae. To illustrate the bio-key regeneration rate varies with n, we tested different bio-key generation rates with the increasing of the extracted minutia numbers. As it is shown in Fig. 12, the bio-key regeneration rate decreases in some degree with the growth of the number n. As n goes from

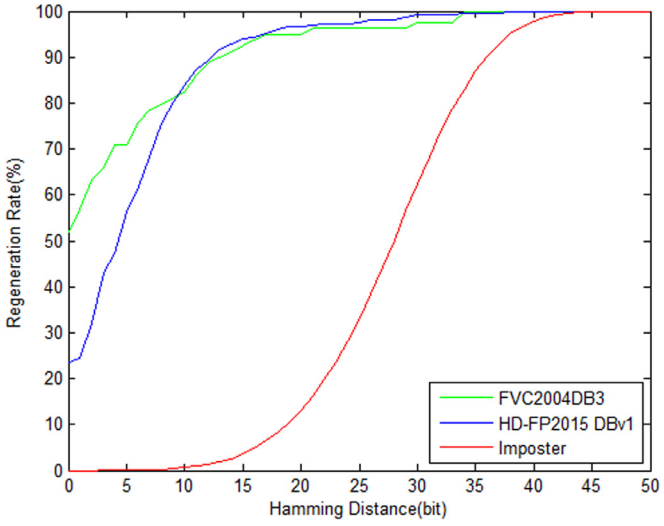

Fig. 11. The bio-key regeneration rate.

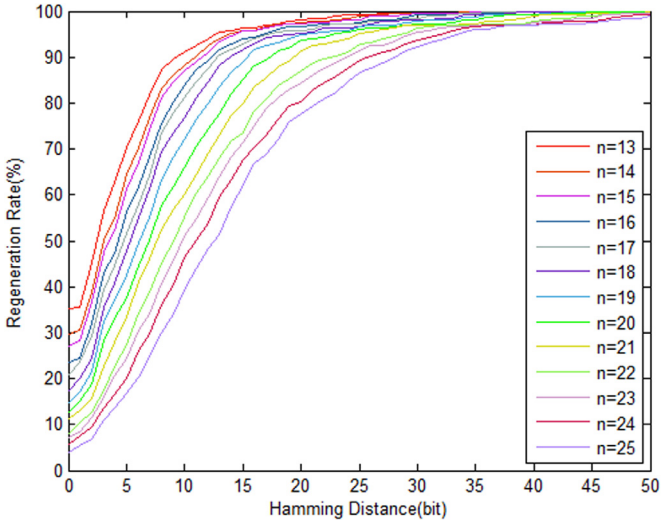

Fig. 12. Bio-key regeneration rates with the number growth of the selected minutiae and predetermined thresholds.

15 to 21, its corresponding bio-key's size will go from 120 to 168 bits, and the bio-key regeneration rate will reach to an ideal value. Thus, it is especially important to obtain both a proper number of the extracted minutiae and an appropriate predetermined threshold for practical applications. Our experiment shows that we can generate a bio-key with its length from 120 to 168 bits at a good bio-key regeneration rate. In addition, we can increase the sizes of our generated bio-keys by choosing the interval codewords with larger sizes, and we can also apply some cryptographic hash function on our bio-key to get a new bio-key having the same size as that of the hash value.

5.3. Effect of the two-layer error correcting

By simulating the noise occurred during transmission, we get the bio-key regeneration rates plotted as Fig. 13 in different error correcting situations. The comparison simulation results are listed in Table 2 with their thresholds are set the same. Both the figure and rate data show that our approach with the two-layer error correcting performs better than the same situations with no error correcting or with only one-layer error correcting.

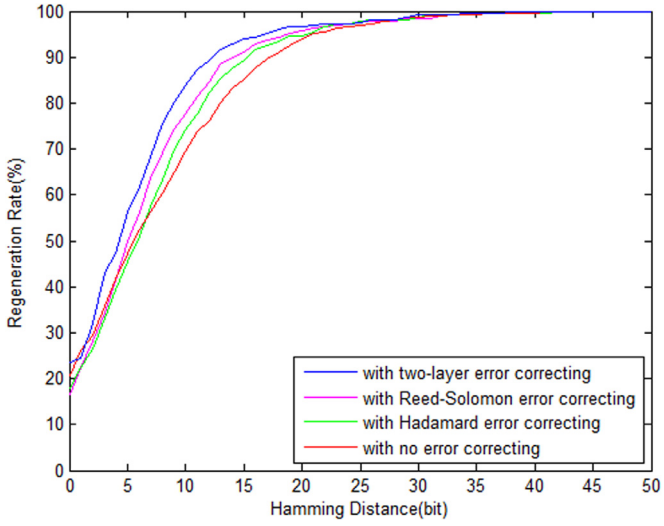

Fig. 13. The effect of error correcting on the regeneration rate with the same thresholds.

Table 2
The bio-key regeneration rates for our approach's 4 versions having two-layer, one-layer or no error correcting.

Our approach's different versions	The bio-key regeneration rate (%)
Two-layer error correcting	92.92
Reed-Solomon error correcting	90.28
Hadamard error correcting	88.63
No error correcting	85.01

5.4. Security analysis

For security of our proposed approach, there exist two key security factors that need to be considered, that is, the security of the fingerprint template and the security of the generated bio-key.

5.4.1. Security of the fingerprint template

Since the bio-key is obtained by the generated interval scheme, the storage of any clearly related information to user's biometric template becomes unnecessary and so the security of user's fingerprint template information is guaranteed to the great extent.

We utilize the relative data which combine the Euclidean distances between the fingerprint minutiae, the differences of the minutia types and the differences of the direction fields to generate the bio-keys, and so once an attacker would obtain the generated intervals and the corresponding codewords, they still cannot recover any minutiae directly from the intervals and the feature distances. The reason is that relative data would not reveal any information about the minutia positions or the topological structure of user's original fingerprint image.

5.4.2. Security of the bio-key

User's original fingerprint bio-key is the concatenation of the codewords of the generated intervals, so once an attacker illegally acquires all the interval codewords, the fingerprint bio-key can be cracked. However, we can immediately update the interval codewords to obtain a totally new fingerprint bio-key. In our experiment, it took an attacker at least 2^{128} attempts to get the correct fingerprint bio-key via the violent cracking when the bio-key size is 128-bit long. As mentioned earlier, the codewords of the generated intervals in our scheme can be randomly selected, and they are equal in length but different from each other. In other word, the dynamic interval codewords will achieve the cancellability of the bio-key and keep that the bio-key will not being exposed.

Table 3
Comparisons with others in average in FAR, FRR, KRGR, R-New, cost and size.

Method	Biometrics	FAR	FRR	KRGR	R-New	Cost (ms)	Sizes
Our approach	Fingerprint	2.58	7.08	92.92	Yes	51.1	$120 \sim 168$
Sheng's[11]	Signature	0	36.10	63.90	No	–	40
Karimian's[14]	ECG	–	–	99.75	Yes	–	$217 \sim 326$
Moosavi's (SEF)[15]	ECG	–	–	–	No	547.5 (diff-CPUs)	128
Seo's in Desktop [18]	Fingerprint	1.65	5.38	–	No	0.018	hash's size
Anees's [20]	Face	0.06	10.02	89.98	Yes	23.1	256
Panchal's [21]	Fingerprint	0.14	0.73	99.27	No	≈ 930	1024
Panchal's [22]	Fingerprint	0.00	6.48	93.52	Yes	1040.71	1274

5.4.3. Comparisons with other recent bio-key generation methods

The two valuable classic metrics for measuring the quality of a biometric authentication algorithm are FAR and FRR. Obviously, both FAR and FRR vary with the preset thresholds. But, since what we mainly concern here is the bio-key regeneration rate, FAR and FRR are not very suitable for measuring the regeneration rate. However, our work including the referred key generation methods applied thresholds to determine whether an almost irreversibly transformed binary string generated from an input biometric pattern matches the previously produced one in the key generation process, hence, the obtained FAR or FRR may affect or reflect the bio-key regeneration rate.

In this subsection, we will compare our approach with other recent valuable work in the aspects including FAR(%), FRR(%) and KRGR(%), R-New, computational cost and the sizes of the generated bio-keys in average. The renewability means that the generated bio-key can be renewed by only changing the transform function or some related parameters but not the corresponding biometric features. For example in our approach, we can renew the bio-key by changing the (dynamic) interval codewords.

We have reviewed a lot of work on biometric-based cryptographic key generation methods, however, quite some of the referenced literatures did not give the regeneration rates of the bio-keys or detailed key generation processes, or only some frameworks about the biometric-based key generations were described.

Here in our comparison Table 3, "–" denotes that there is no experimental data or results in the corresponding literatures, "547.5 (diff-CPUs)" denotes that 547.5 is the average time cost in the computational environments of different types of processors, and the "hash's size" means that the size is the same as that of the used hash function's value.

As shown in Table 3, our approach's bio-key generation rate is higher than that of Sheng et al.'s and Anees et al.'s, while it is lower than that of Karimian et al.'s and Panchal et al.'s. Four approaches including ours have dynamic bio-key generation mechanisms, that is, their bio-keys can be renewed by changing the transform algorithms or some parameters without extracting or inputting a new biometric feature (sample). Our approach costs much less time than Moosavi et al.'s and Panchal et al.'s two approaches, while it takes a little more time to generate a bio-key than Anees et al.'s and Seo et al.'s in the desktop environment. In addition, the size of the generated bio-key by our approach is smaller than the sizes of the bio-keys generated by Karimian et al.'s, Anees et al.'s, or Panchal et al.'s.

In addition, compared with Anees et al.'s biometric key generation framework, though our approach also needs to store the auxiliary data for the bio-key regeneration, it will not expose the fingerprint minutiae if the auxiliary data is leaked. While if their stored auxiliary perturbation vector $P = \{\rho_1||\rho_2||\cdots||\rho_{59}\}$ is illegally disclosed, then the quantized features f'_{iq} may be leaked as $f'_{iq} = (\rho_i + 1) \bmod 8$ through $\rho_i = [\rho_{i,1}, \rho_{i,2}, \rho_{i,3}] = (f'_{iq} - 1) \bmod 8$, and furthermore, some LBP pattern features f_i may be exposed through the inverse operations of the Eqs. (5)–(7) in [20]. As Seo et al. described in their work [18], that their BB-KDF could derive a biometric key in much less time than others, but they assumed that as many minutiae as before could be exacted at the key regeneration stage, and their method's regeneration rate was not given.

It is worth mentioning that, for all the bio-key generation approaches cited in Table 3, only our approach has been practically experimented to generate a bio-key.

Based on the comparisons above, we know that our approach provides a pretty acceptable bio-key regeneration rate and computational cost since the user can regenerate his bio-key under the baseline probability in a reasonable short time. Note that our approach will take much time to extract and process multiple training biometric samples from the same input biometric feature to obtain the more accurate biometric features while the regeneration procedure costs much less time actually. Therefore, we can reduce the running time of our approach by controlling the number of the training biometric samples. To generate longer bio-key, we can expand the size of our bio-key by making some minor changes or operations, such as coding the feature interval with longer codewords. In addition, the generated bio-key itself is also renewable so that the key can be kept secure enough for cryptographic applications. Besides, we can employ or develop a more accurate biometric (fingerprint) feature extraction algorithm for our approach to improve the bio-key regeneration rate.

However, it must be mentioned, since the biometric key generation methods given in our referenced literatures were experimented in different environments, our comparison table about the bio-key generation rates and the related cost time may not really reflect their performances.

6. Four application scenarios

Our bio-key generation approach can be considered to be applied in the following scenarios. ***Digital Identity*** Our bio-key generation approach can be applied for identity authentication. A user can prove his identity by generating his bio-key since only his successfully regenerated bio-key from his genuine biometric feature can match with his original biometric bio-key. There are two advantages to employ our bio-key generation approach for digital identity: one is that in case a user's bio-key digital identity is illegally used by others, he can renew his bio-key identity by changing the interval codewords, and the other is that the user's bio-key digital identity will not exposed the user's real biometric features even if his bio-key would be illegally used by some attacker. ***Digital Wallets*** In a cryptocurrency or blockchain system, all users must set up their own digital wallets using the wallet systems to securely keep their private keys. Currently, almost every digital wallet needs a user to record a mnemonic word which is consisted of at least twelve words and is too long for his memory. Hence, the mnemonic word should be recorded in a paper, a mobile phone or other physical mediums. Of course, one can choose to keep his private key instead of the corresponding mnemonic word. But for security, the private key used in a digital wallet is generally set to be a 64-bit hexadecimal number with no meaning, and so it should

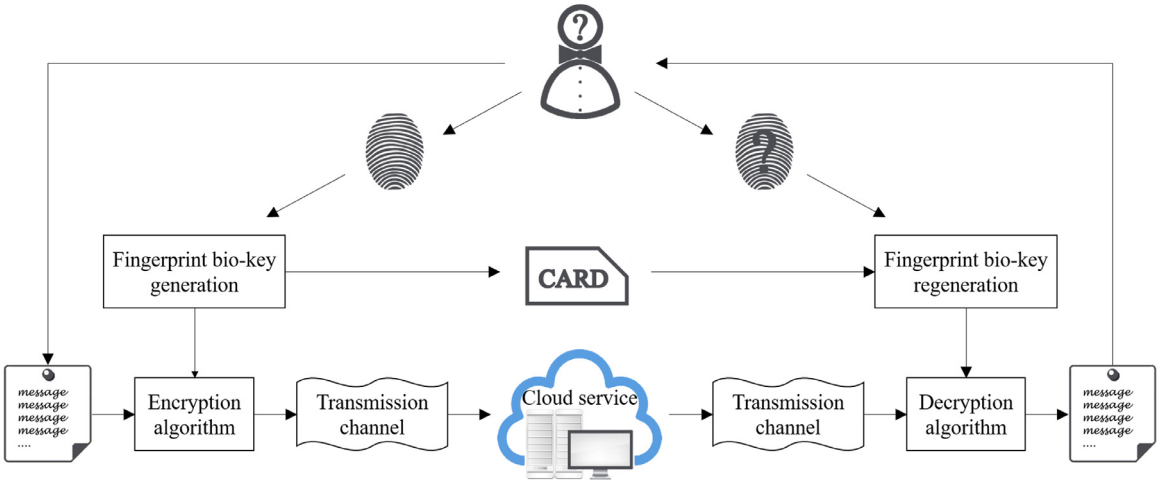

Fig. 14. An application scenario in cloud data encryption using our approach.

also be recorded or stored in a physical medium. Therefore, such digital wallet systems are not very convenient for the users who do not trade frequently. We can develop a biometric based digital wallet system which employs our approach to generate a bio-key used as a user's private key. Such biometric private key needs not to be kept in the digital wallet since it can be regenerated by extracting the user's biometric features when the user wants to enter his digital wallet. In addition, such biometrics-based wallet systems will save us from having to generate mnemonic words. ***Data Encryption*** Nowadays, a large amounts of data are transmitted on public networks everyday and they are faced with the privacy and security issues. The bio-key generated using our approach is not only convenient for acquisition and recognition but also has the stability and uniqueness which a general digital key or physical identity object does not have for a specified user. Hence, our bio-key generation approach can be applied in symmetric encryption algorithms. A user can transmit his secret data to the cloud services for secure storage or for some cloud computing services by encrypting his data using his unique bio-key without needing to remember any password or worrying about the loss of his private key. The application of our approach in cloud data encryption can be described as Fig. 14. In addition, in case his bio-key becomes compromised, he can update his bio-key by changing the codewords of the generated intervals.

ID-based Cryptographic Algorithm In particular, our bio-key approach can be applied for ID-based cryptographical algorithms. For example, in an ID-based signature, the bio-key generated by our approach from the signer's biometric feature can be used as his public identity ID_{bio}. The key generation center (KGC) who has a master public and secret key pair (mpk, msk) can use $ID_{bio}$and msk to generate the signer's private key $d_{ID_{bio}}$. Then, the signer can use his private key $d_{ID_{bio}}$ to make a signature on the message m: $sig_{bio} = \mathbf{Sign}(m, d_{ID_{bio}})$. Later, any one not only can verify the signature sig_{bio} using the signer's ID_{bio} and mpk, but also can simultaneously authenticate the signer's genuine identity since only the signer can generate the identity ID_{bio} from his biometric feature.

7. Conclusion

In this work, we propose a new biometric bio-key generation approach based on the generated interval scheme with a two-layer error correcting technique. We design and simulate the realization interfaces and made the experimental tests based on the two fingerprint databases.

Our experiment results prove that this bio-key generation approach shows a better security performance and provides an acceptable bio-key regeneration rate at a low computational cost. The bio-key size of our approach is adjustable between 120 to 168 bits long for the extracted minutia numbers from 15 to 21, and the fingerprint bio-key regeneration rate is up to 91.14% for the database FVC2004 DB3. However, we can increase the generated bio-key's size by choosing longer interval codewords or applying some hash function on the bio-key to generate a new bio-key of a larger fixed length. Moreover, the positive effect of the two-layer error correcting technique is also tested out and it indeed illustrated that this technique can efficiently insure the personal data against some unpredictable interference occurs during transmission. Finally, we have given four possible practical scenarios in which our biometric key can be applied and hope it can inspire others to some extent.

However, our approach exists some limitations. Compared with the minutia alignment-free method for fingerprint template protection presented in [27], our generated bio-key is constructed by arranging the distance intervals which may expose some biometric information, but we can set the codewords of the intervals dynamically and isometrically to protect the security of the fingerprint and also achieve the cancellability of the fingerprint bio-key. Besides, we have to extract multiple training image samples of a fingerprint to get a stable minutia set and it makes our approach take much time to generate a bio-key. Hence, in our future work, a more precise minutia extraction algorithm is urgently needed to be developed for our approach since our bio-key information is mainly depended on the extracted minutia information. In addition, a formal mathematical security model needs to be set up for cryptographic bio-key generation algorithms. Naturally, Our bio-key generation approach on fingerprint is also feasible for other biometrics such as finger vein and face.

Declaration of Competing Interest

The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

CRediT authorship contribution statement

Peiyi Wang: Conceptualization, Writing - review & editing. **Lin You:** Conceptualization, Formal analysis, Writing - original draft. **Gengran Hu:** Formal analysis, Writing - review & editing. **Liqin Hu:** Formal analysis, Validation. **Zhihua Jian:** Validation. **Chaoping Xing:** Formal analysis.

Acknowledgements

This research is partially supported by the Key Program of the Nature Science Foundation of Zhejiang province of China (No. LZ17F020002) and the National Natural Science Foundation of China (No. 61772166).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.patcog.2020.107733.

References

[1] G. Tomko, C. Soutar, G. Schmidt, Fingerprint controlled public key cryptographic system, U.S. Patent 5832091 (1996) US5832091A.
[2] T. Ong, W. Khoh, A. Teo, Dynamic handwritten signature verification based on statistical quantization mechanism, in: Proc. IEEE Int. Conf. Comput. Eng. Technol., vol.2, 2009, pp. 312–316, doi:10.1109/ICCET.2009.128.
[3] C. Rathgeb, A. Uhl, Context-based texture analysis for secure revocable iris biometric key generation, in: Proc. 3rd IET Int. Conf. Imaging Crime Detect. Prev., vol. 1, 2009, pp. 1–6, doi:10.1049/ic.2009.0229.
[4] C. Rathgeb, A. Uhl, Privacy preserving key generation for iris biometrics, in: Proc. IFIP Int. Conf. Commun. Multimed. Secur., vol. 6109, Springer, 2010, pp. 191–200, doi:10.1007/978-3-642-13241-4_18.
[5] Y. Dodis, L. Reyzin, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, in: Advances in Cryptology- EUROCRYPT 2004, vol. 3027, Springer, 2004, pp. 523–540, doi:10.1007/978-3-540-24676-3_31.
[6] F. Zhang, D. Feng, Z. Sun, An iris authentication scheme based on fuzzy extractor, J. Comput. Res. Dev. 45 (6) (2008) 1036–1042.
[7] R.Á Mariño, F.H. Álvarez, L.H. Encinas, A crypto-biometric scheme based on iris-templates with fuzzy extractors, Inf. Sci. 195 (13) (2012) 91–102, doi:10.1016/j.ins.2012.01.042.
[8] W. Sheng, G. Howells, M. Fairhurst, et al., Reliable and secure encryption key generation from fingerprints, Inf. Manag. Comput. Secur. 20 (3) (2012) 207–221, doi:10.1108/09685221211247307.
[9] S. Wang, J. Hu, Design of alignment-free cancelable fingerprint templates via curtailed circular convolution, Pattern Recognit. 47 (3) (2014) 1321–1329, doi:10.1016/j.patcog.2013.10.003.
[10] L. Leng, A. Teoh, Alignment-free row-co-occurrence cancelable palmprint fuzzy vault, Pattern Recognit. 48 (7) (2015) 2290–2303, doi:10.1016/j.patcog.2015.01.021.
[11] W. Sheng, S. Chen, G. Xiao, et al., A biometric key generation method based on semisupervised data clustering, IEEE Trans. Syst. Man Cybern. 45 (9) (2015) 1205–1217, doi:10.1109/tsmc.2015.2389768.
[12] S. Partheeba, N. Radha, Fingerprint bio-Crypto key generation using scale invariant feature transform (SIFT), Int. J. Comput. Appl. 153 (8) (2016) 35–41, doi:10.5120/ijca2016912129.
[13] D. Lavinia, H. Gerhard, User-centric key entropy: study of biometric key derivation subject to spoofing attacks, Entropy 19 (2) (2017) 70–91, doi:10.3390/e19020070.
[14] N. Karimian, Z. Guo, M. Tehranipoor, D. Forte, Highly reliable key generation from electrocardiogram (ECG), IEEE Trans. Biomed. Eng. 64 (6) (2017) 1400–1411, doi:10.1109/TBME.2016.2607020.
[15] S. Moosavi, E. Nigussie, M. Levorato, et al., Low-latency approach for secure ecg feature based cryptographic key generation, IEEE Access 6 (2018) 428–442, doi:10.1109/ACCESS.2017.2766523.
[16] G. Verma, A. Sinha, Securing information using optically generated biometric keys, J. Opt. 18 (11) (2016) 1–12, doi:10.1088/2040-8978/18/11/115701.
[17] G. Verma, M. Liao, D. Lu, et al., An optical asymmetric encryption scheme with biometric keys, Opt. Lasers Eng. 116 (11) (2019) 32–40, doi:10.1016/j.optlaseng.2018.12.010.
[18] M. Seo, J.H. Park, Y. Kim, et al., Construction of a new biometric-based key derivation function and its application, Secur. Commun. Netw. 2018 (2018) 1–14, doi:10.1155/2018/6107912.
[19] Z. Wu, L. Tian, P. Li, et al., Generating stable biometric keys for flexible cloud computing authentication using finger vein, Inf. Sci. 433–434 (2018) 431–447, doi:10.1016/j.ins.2016.12.048.
[20] A. Anees, Y. Chen, Discriminative binary feature learning and quantization in biometric key generation, Pattern Recognit. 77 (2018) 289–305, doi:10.1016/j.patcog.2017.11.018.
[21] G. Panchal, D. Samanta, A novel approach to fingerprint biometric-based cryptographic key generation and its applications to storage security, Comput. Electr. Eng. 69 (2018) 461–478, doi:10.1016/j.compeleceng.2018.01.028.
[22] G. Panchal, D. Samanta, S. Barman, Biometric-based cryptography for digital content protection without any key storage, Multimed. Tools Appl. 78 (19) (2019) 26979–27000, doi:10.1007/s11042-017-4528-x.
[23] U.S. Tomar, A. Vidwans, P. Sharma, Fingerprint recognition by hybrid optimization based on minutaies distance and pattern matching, in: Proc. IEEE. Int. Conf. Signal Process. Commun. Power Embed. Sys., 2017, pp. 2047–2051, doi:10.1109/SCOPES.2016.7955808.
[24] P. Zhang, J. Hu, C. Li, et al., A pitfall in fingerprint bio-cryptographic key generation, Comput. Secur. 30 (5) (2011) 311–319, doi:10.1016/j.cose.2011.02.003.
[25] J. Massey, Shift-register synthesis and BCH decoding, IEEE Trans. Inf. Theory 15 (1) (1969) 122–127, doi:10.1109/TIT.1969.1054260.
[26] D. Sadhya, S.K. Singh, Design of a cancelable biometric template protection scheme for fingerprints based on cryptographic hash functions, Multimed. Tools Appl. 77 (12) (2018) 15113–15137, doi:10.1007/s11042-017-5095-x.
[27] M. Sandhya, M. Prasad, k-Nearest neighborhood structure(k-NNS) based alignment-free method for fingerprint template protection, in: Proc. IEEE Int. Conf. Biom., 2015, pp. 386–393, doi:10.1109/ICB.2015.7139100.

Peiyi Wang is a Ph.D. candidate majored in information security and biometric recognition at Hangzhou Dianzi University. Her research interest is mainly about biometric encryption technology and she now studies in the cryptographical key generation based on biometrics.

Lin You is a professor of the School of Cyberspace Security at Hangzhou Dianzi University and is also the director of the Institute of Cryptography and Information Security. His research interests include cryptography, biometric recognition and their combinations & applications. He is a member of IEEE, IACR and CACR, respectively.

Gengran Hu is a lecturer of the School of Cyberspace Security at Hangzhou Dianzi University. His research interests include lattice-based cryptography, blockchain and their applications. In addition, he is a member of CACR.

Liqin Hu received the Ph.D degree in mathematics from the Nanjing University of Aeronautics and Astronautics, Nanjing, China. She is a lecturer of the School of Cyberspace Security at Hangzhou Dianzi University. Her research interests include cryptography and coding theory.

Zhihua Jian received the Ph.D degree in signal and information processing from the Nanjing University of Post and Telecommunication in 2008. Now he is an association professor at the School of Communication Engineering, Hangzhou Dianzi University. His research interests include voice conversion, speaker recognition and anti-spoofing.

Chaoping Xing received his Ph.D. degree in 1990 from University of Science and Technology of China. He joined University of Essen, Germany as an Alexander von Humboldt fellow from 1993 to 1995. From 1998 to 2007, he worked in National University of Singapore. From 2008 to 2019, he worked as a tenure professor in National University of Singapore. Since 2020, he has been with Shanghai Jiao Tong University as a full professor. His research interests include coding theory and cryptography.