ANDROID SECURE STORAGE APPLICATION USING FUZZY VAULT-BASED KEY BINDING

Sanaa Ghouzali[1]*, *Haya Altuwaijri*[1], *Maryam Lafkih*[2], *Mohammed Al-Goblan*[3], *Wadood Abdul*[3]

[1] Information Technology Department, College of Computer and Information Sciences, King Saud University, Riyadh
[2] SmartiLab, National School of Engineering Sciences, Rabat
[3] Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh
[1,3] Saudi Arabia, [2] Morocco

* Corresponding Author: e-mail: sghouzali@ksu.edu.sa

Abstract: In the last decade, mobile devices become a storage place for highly sensitive information such as authentication credentials, bank account information, etc. However, smartphones are more likely to be lost or stolen than desktops or even laptops, putting user data at a high risk of being compromised. Malware and legitimate vulnerable applications can also be a threat to data disclosure. Therefore, this work focuses on preserving the confidentiality of the information stored in mobile devices from unauthorized access and specifically in the Android platform using a fuzzy vault-based key binding algorithm. The proposed approach aims to conceal the cryptographic key with the user's fingerprint such that only the genuine user will be authorized to access the secret key.

Key words: Android; Data Security; Biometric cryptosystems; Fuzzy Vault; Fingerprint authentication.

1. INTRODUCTION

With the advance in technology, the use of mobile devices has grown exponentially this last decade as compared to other devices such as desktops, laptops, and tablets. People are using their devices to perform highly sensitive transactions such as accessing banking accounts and online shopping. Henceforth, highly sensitive information such as authentication credentials, bank account information, photos and videos are being stored in mobile devices. The confidentiality and privacy of this data is becoming a critical issue for mobile users. Appropriately securing mobile data includes two dimensions that need to be considered [1]. First, protect

data from physical threats, in case the device is lost or stolen. Second, protect data from software threats by preventing improper access to the data. Software threats can be solved using cryptography. However, cryptographic techniques rely on the assumption that only the legitimate user knows the cryptographic key, hence, maintaining the secrecy of keys is a big challenge. An attacker who gains access to the device can extract the key if it is locally stored. Thus, cryptography is vulnerable to physical threats.

The biometric cryptosystem approach is a promising solution that can be used to secure data against software and physical attacks since data is encrypted, and the cryptographic key is never stored in the device [2]. The aim of this approach is to protect sensitive data (e.g., encryption key) with the user's biometric data such that only the genuine user will be authorized to access the secret by providing the authentic biometric. In this case, the secrecy of the key will be guaranteed since biometrics cannot be easily forged, duplicated, or shared [3]. The aim of this paper is to propose a secure data encryption for mobile device storage using biometric cryptosystem based on key binding approach. The encryption key is protected using a fingerprint-based fuzzy vault algorithm. The scope of this application is limited to Android operating system because it has the highest market share worldwide on mobile devices with an 84% share in the first quarter of 2021 [4].

This paper is organized as follows. Related work is presented in Section 2. In Section 3, the fingerprint-based fuzzy vault preliminaries are presented. The application design and implementation are shown in Section 4. Next, the application evaluation is discussed in Section 5. Finally, conclusion and future work are drawn in Section 6.

2. RELATED WORK

According to Mobile Security Project under The Open Web Application Security Project (OWASP) [5] that defines the top 10 mobile risks, insecure data storage is listed as one of the main security issues in smartphones since sensitive information can be revealed if it is not protected carefully. There are two types of threats in Android data storage, physical threats and software threats [1]. To protect the mobile device against data disclosure, data can be encrypted. Android provides two types of encryption systems, Full Disk Encryption (FDE) and Key-Chain, and both require a strong password to prevent illegitimate access to the data. However, it is very likely for the user to choose a weak password that is easy to remember.

The use of biometrics in securing user authenticity has grown in the last few years, and many smartphone manufacturers have adopted the use of biometric authentication in applications such as: phone lock, mobile payment, and mobile banking transactions [6]. Other applications merge cryptography and biometrics, called Biometric cryptosystems, to achieve high security and user convenience simultaneously. Most importantly, these methods can be applied to biometric template protection and cryptographic key generation ([7, 8]). Several biometric

template protection approaches have been proposed using Biometric cryptosystems such as Fuzzy extractors, Fuzzy vault, and Fuzzy commitment [9].

In [10], the authors implemented an application for fingerprint authentication in mobile devices based on the fuzzy extractor. The scheme has shown to be more secure to protect the biometric template from unauthorized access to the mobile storage. In [11], the authors used Reed-Solomon (RS) encoding to generate a codeword from the user's fingerprint and then generated a key from the codeword. Key decryption is fulfilled using the user's biometric data after verifying the user's authenticity. In this approach, neither templates nor keys are required to be stored. In [12], the authors have employed the template formation method to combine the cryptographic key with the template features by using Delaunay triangulation. After extracting the minutiae points and generating the random key, the key is combined with the minutiae points in one set. In order to remove the location information, the combined set is transformed using Delaunay triangulation. Another way to transform the template is proposed in [13], where the authors have used two generated keys to relocate the minutiae points of the user. To enhance the security of the approach, two more keys are generated to rotate and translate the previously relocated minutiae point with the respect to the origin. In [14], a location-based encryption model using a fuzzy vault scheme has been proposed to secure data in mobile devices.

3. FINGERPRINT-BASED FUZZY VAULT PRELIMINARIES

A key binding-based fuzzy vault algorithm is developed to secure the encryption key by applying the user's fingerprint. The algorithm requires the detection of prominent singular points, known as minutiae. Most of minutiae extraction algorithms attach a minutia with four features: coordinates (a, b), orientation (θ), and type (bifurcation or termination) [15]. The algorithm generates a Helper data using the minutiae and the encryption key. Only the Helper data will be stored in the device. Key binding-based fuzzy vault algorithm consists of two processes: vault encoding and vault decoding.

3.1. Vault encoding

The vault encoding can be summarized as follows [16]:

1) Given fingerprint template T, minutiae points $X = \{m_i\}_{i=1}^{r}$ are extracted, where r is the number of minutiae points in T. Only the top-quality minutia points are selected.
2) The attributes a, b, θ are quantized and represented as bits string. The first 16 bits are selected and converted to a single integer number, and added to X.
3) The secret K is partitioned c_0, c_1, c_2, ..., c_n, where n is the degree of polynomial P. Then, the partitioned K is encoded into P of degree n, by considering them as coefficients of P (i.e., $(P(x) = c_n x^n + ... + c_0)$.
4) The polynomial $P(x)$ is evaluated at minutia points M^T by performing the locking set $\{(x_j, P(x_j))\}_{j=1}^{r}$.

5) A set of Chaff points is generated. The selected Chaff point should be higher than δ (the most significant number of minutiae points after quantization). The Chaff point should not lie on the polynomial P (i.e., Chaff point (x, y) where $P(x) \neq y$). The Chaff points set is defined as $C = \{(x_k, y_k)\}_{k=1}^{s}$, where s is the number of Chaff points.
6) The union of L and C is denoted as Vault $V = L \cup C$, which is randomly reordered and stored in the device.

3.2. Vault decoding

The vault decoding is summarized as follows [16]:

1) Given fingerprint query template T^Q, the same steps 1) and 2) in the vault encoding phase are followed to extract the minutiae points X^Q.
2) Calculate the most significant number in X^Q, which is δ.
3) Retrieve the vault V and remove Chaff points such as: the i^{th} element of V is marked as Chaff point if it is higher than the most significant number δ.
4) Elements of V, which have a corresponding minutia in X^Q, are considered as unlocking set L.
5) Construct a polynomial of degree $n + 1$ to extract the key. If the number of elements in $L < (n + 1)$, it results in an authentication failure. Otherwise, construct a polynomial P using Lagrange interpolation.
6) Extract the coefficients of P that constitute the key K.

4. APPLICATION DESIGN AND IMPLEMENTATION

The proposed application aims to secure the storage model in the Android environment. The encryption key is protected from eavesdropping using the fingerprint-based fuzzy vault algorithm described previously. Figure 1 shows the component view of the application in the Android environment.

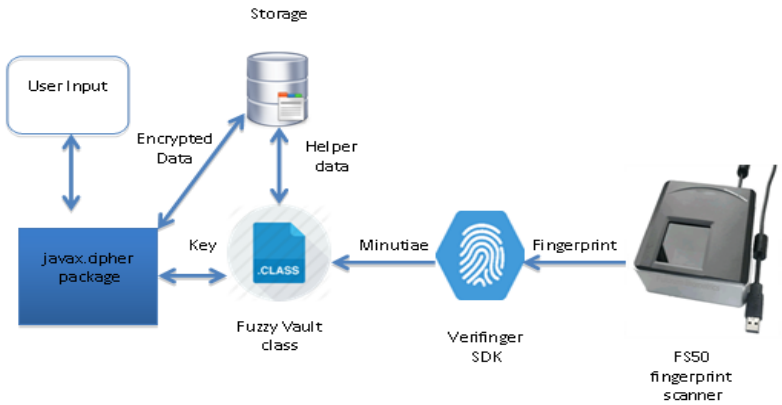

Fig. 1. Application Architecture Design.

In the key encryption phase, after acquiring a fingerprint image, which is converted to a binary image, the minutiae features are extracted from the binarized images. These features are then used to bind the cryptographic key by applying a fuzzy vault encoding algorithm that produces a helper data to be stored. Note that neither the cryptographic key nor the fingerprint features are stored. In the decryption phase, the helper data is retrieved from the device storage. After acquiring the user's fingerprint image, minutia points are extracted. These minutia features with the retrieved helper data are used in a fuzzy vault decoding algorithm to recover the cryptographic key. To better understand the entire application architecture, these components are explained in detail in this section.

4.1. Fingerprint Minutia Extraction

In this study, *FS50* fingerprint scanner is used to acquire the user's fingerprint image because accessing the user's fingerprint from the mobile built-in scanner is restricted by Android. To extract the fingerprint minutia features, *Verifinger 8.0* software development kit (SDK) is used in this work [17]. *Verifinger* SDK for biometric systems includes Fingerprint Matcher and Extractor components. The latter provides classes and methods (*EnrollFingerFromImage.java*) to extract fingerprint features, including a and b coordinates, angle θ, and quality. The quality of the minutia is used to determine the top 16 quality minutia points to be handled. Then the features a, b, and θ, are quantized to a one single value. The quantization process is done as follows: (1) convert a, b, and θ to binary, (2) store the binary bits in an array, (3) select the first 16 bits, and (4) convert the 16 bits to an integer. This process will produce a single value ranging from 0 to *65535* that represents the three values (a, b, θ). The aim of converting the fingerprint features to a single value is to evaluate the polynomial on these values.

4.2. Data Encryption-based Secrecy Code

Android provides encryption algorithms that permit developers to encrypt data in their applications to improve their security using '*javax.cipher*' package. In this work, we utilize *Secrecy* open-source code, available on GitHub[1], which satisfies the requirements of Android encryption [18, 19]. This code provides an encryption solution for securely storing the files in the mobile using Advanced Encryption Standard (AES) symmetric block cipher with the Counter-mode (CTR) of operation. The user can create different folders, named vaults, and each folder is protected by a passphrase. The user can add any type of files to these folders and all files in these folders will be encrypted. The encryption key is generated randomly for each vault using the *KeyGenerator* class that is available under *javax.crypto* package. A secret key is randomly derived from the user's passphrase and a salt value using a pseudorandom number generator method (*generateSecret()*). The encryption key is

[1] https://github.com/SecrecySupportTeam/secrecy

wrapped by the secret key derived from the passphrase. The wrapping process is done using the *Cipher* class that is available under *javax.crypto* package.

For decryption, the encoded key is retrieved from the device storage. The user is required to enter the passphrase in order to derive a secret key which is used then in the unwrapping process to decode the encryption key. If the key from the passphrase is wrong, the unwrapping method will throw an exception, and the message (*Passphrase is wrong!*) will appear to the user.

4.3. Data Encryption-based Secrecy+ Code

Instead of wrapping the encryption key using a passphrase, the proposed scheme uses a fingerprint-based fuzzy vault algorithm to protect the encryption key. The modified code implemented in this work, named *Secrecy+*, consists of two main processes, key encoding and key decoding.

4.3.1. Key encoding

The key encoding is implemented using the Apache Commons Math library that provides mathematics and statistics solutions. Using this library, we generate random polynomial and set the encryption key as its coefficients. The size of the AES encryption key in this work is 128−bits. Then we evaluate the polynomial on minutia point. The output of the evaluation is saved in *polynomial_values* global variable. The *AddChaffPoints()* method generates a random number, which is above the largest number in minutia points. This random number will be considered as minutia point (C). Next, we generate another random number that will be considered as the output of evaluating Chaff point (C') on polynomial (P) such that $P(C) \neq C'$. In the end, we call the *shuffleArrayList()* method that randomizes the ordering of genuine and Chaff points. In this case, we can differentiate Chaff points and genuine points in key decoding. The flowchart of the key encoding process is presented in Figure 2.

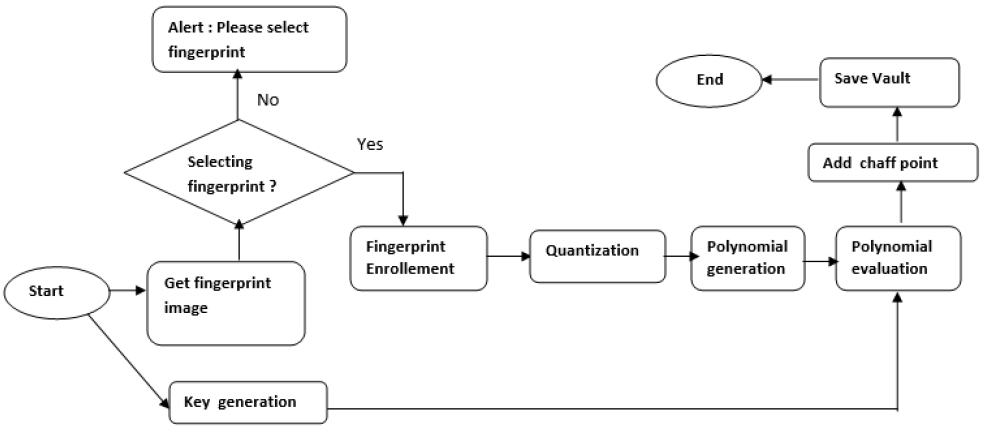

Fig. 2. Key encoding flowchart in Secrecy+ Application.

4.3.2. Key decoding

First, Chaff points are removed from the retrieved vault using *removeChaffPoints()* method. Then, we compare minutia points from the vault with minutia points extracted from the query fingerprint using *comparison()* method. In case of a mismatch, an alert message (*Verification failed!*) is shown to the user. Otherwise, we use the *PolynomialFunctionLagrangeForm* class from Apache Commons Math library to recover the key. This class applies Lagrange polynomials mathematical theory for polynomial interpolation. Giving the genuine points that consist of minutia points (x) and polynomial values ($P(x)$), *PolynomialFunctionLagrangeForm* will reconstruct the polynomial (P). The key can be retrieved by calling the *getCoeffecient()* method. The flowchart of the key decoding process is presented in figure 3.

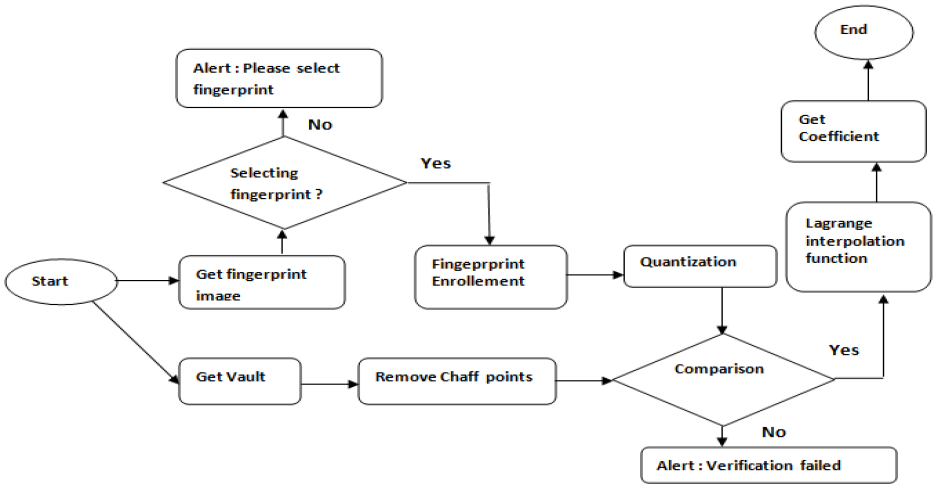

Fig. 3. Key decoding flowchart in Secrecy+ Application.

5. APPLICATION EVALUATION

In this section, the security analysis of the proposed scheme is investigated to determine the amount of information an attacker can gain when the vault is leaked. The brute-force attack's feasibility is also analysed. Moreover, a comparison is provided between *Secrecy*, the original data encryption-based passphrase code, and *Secrecy+*, the improved version using the Fuzzy vault algorithm. In addition, the application's efficiency in Android device is presented, followed by a comparison with other solutions.

5.1. Information leakage from Vault

Security analysis will be conducted in case the attacker can gain access to the vault in storage. The security analysis of the proposed Android storage application

will be evaluated using the following parameters: 1) the key size, 2) the polynomial degree, and 3) the number of added Chaff points. Our scheme is designed to secure a secret key of length 8(n + *1*) bits, where n is the encoding polynomial degree. In this work, the key size is chosen to be 128−bits, and hence the degree of the polynomial is 15. Since each byte of the key is considered a coefficient of the polynomial, the scheme's security is in adding Chaff points that conceal the genuine minutia points, thus making it hard for the attacker to construct the correct polynomial. The number of Chaff points in the vault is chosen to be larger than the genuine points as it is recommended in [16]. However, a trade-off between the security of the vault and its storage requirements should be considered to determine the number of Chaff points.

Suppose the attacker gains access to the stored vault which consists of (x, $P(x)$), where x is the minutia point after quantization. Since the quantization process is a one-way function, the attacker cannot reconstruct the real fingerprint template from the vault. Moreover, $P(x)$ does not provide any information regarding the polynomial. Also, since the Chaff and genuine points are reordered randomly in the vault, there is no way to differentiate the Chaff points from genuine points if the attacker does not know the user's input (fingerprint).

5.2. Brute-force attack

The attacker might attempt a brute-force attack on the proposed scheme by trying to decode the secret using all combinations of (n+*1*) which is equal to r = *16* points in the vault. He should reconstruct the polynomial for each combination and retrieve its coefficients, which is the key and verify its correctness. The analysis will provide a calculation of the time needed by an attacker to successfully decode the key, as presented in [16]. Moreover, the min-entropy will be calculated to determine the attack feasibility to decode the vault. The min-entropy has been used as a security analysis for the fuzzy vault in [20]; it is merely representing the minimum possible trials an attacker has to perform in a brute force attack.

In order to prevent this attack, we will examine the effect of adding several numbers of Chaff points that are larger than genuine points as recommended in [16]. The strength of the vault and the brute force calculations are illustrated in Table 1 which shows the number of Chaff points s, and possible combinations that are calculated as $\binom{r+s}{r}$. The min-entropy of the vault is calculated using $\log_2\binom{r+s}{r}$. It can be seen that when the number of Chaff points increases, the number of possible combinations also increases, which increases the min-entropy of the fuzzy vault. In our implementation, we use 20 Chaff points; therefore, the attacker should try to decode the key 2^{32} times which is equal to 4,294,967,296 trials. For a processor that needs one second for million trials, the computational time is roughly equal to one hour to achieve success. Henceforth, it is required to add more Chaff points to foil a brute-force attack. However, a trade-off between the security of the vault and the

application performance should be considered to determine the adequate number of Chaff points.

Table 1. Min-entropy of the Vault for different numbers of Chaff points

# Chaff points	# Possible Combinations	Min-Entropy	# Alternative Keys	Time required at 10^3 trials/s (~)	Time required at 10^6 trials/s (~)
17	$5.65x10^8$	29	2^{29}	6 days	8 minutes
20	$7.30x10^9$	32	2^{32}	49 days	1 hour
30	$9.91x10^{11}$	39	2^{39}	17 years	6 days
40	$4.16x10^{13}$	45	2^{45}	951 years	1 year
50	$8.55x10^{14}$	49	2^{49}	15854 years	15 years

5.3. Comparing *Secrecy+* to *Secrecy* and Other Solutions

The original *Secrecy* application relies on the user's passphrase to conceal the encryption key. However, there are no restrictions on the passphrase entered by users; even a single character is accepted. For this comparison, we suppose that there are minimum requirements for the passphrase as NIST recommends it in [21]. Henceforth, passphrases will be created with 30 bits of min-entropy [20]. While in our case, we can upgrade the min-entropy by adding Chaff points without requiring users to remember too long, complex, and unmemorable passphrases. Moreover, we examined the impact of applying a fuzzy vault algorithm in Android devices in terms of speed. The required time for key encoding and key decoding is 2.246 seconds using *Secrecy* compared to 5.016 seconds using *Secrecy+*. We can infer from the obtained results that applying the fuzzy vault algorithm provides better security while producing minor speed degradation.

Most of the other provided security solutions such as Android FDE, Android Key Chain, and *Secrecy* depend on the strength of the chosen passphrase and such are vulnerable to brute force attacks and social engineering attacks. The proposed scheme enhances information security since it protects data against software attacks because data is encrypted, and against physical attacks because the cryptographic key is not stored in the device, only an auxiliary data is stored. Moreover, it provides the user's convenience since it depends on biometric characteristics; no need to remember complex and lengthy passwords, or hold verification token. Besides, as technology advances and more powerful processors appear, the user will not notice the speed degradation when using the fuzzy vault-based security algorithm.

6. CONCLUSION

Cryptography is the main procedure used to secure the device's data. However, maintaining the confidentiality of the key remains a challenge. The key is vulnerable to physical attack if stored in the device even when it is hashed or encrypted. Moreover, since mobile devices can be easily lost or stolen, storing the key in them

is not recommended. Hence, we proposed a solution that protects the storage of the key in the device. The proposed solution is implemented using biometric cryptosystems based on the key binding approach. Specifically, it uses a fuzzy vault-based key binding algorithm to conceal the cryptographic key with the user's fingerprint. To validate the proposed application design, we have employed an open-source code that implements the encryption process, and we have applied a fuzzy vault algorithm for the key coding and decoding. Furthermore, we have followed a quantitative evaluation to examine the information leakage from the vault and the applicability of the brute-force attack. We deduced a number of recommendations based on the analysis results in order to prevent these shortcomings. Moreover, we have investigated the efficiency of applying the proposed scheme in Android devices and concluded slight degradation of performance which can be circumvented with the advanced technology in the future. The proposed solution protects the key from being retrieved from the internal storage only. For future work, this solution would be enhanced by eliminating the risk of gaining the key from the RAM as well.

REFERENCES

[1] H. Altuwaijri, S. Ghouzali. Android data storage security: A review. *Journal of King Saud University - Computer and Information Sciences*, Vol. 32, No. 5, 2020, pp. 543–552, http://doi.org/10.1016/j.jksuci.2018.07.004.

[2] Y. J. Lee, K. Bae, S. J. Lee, K. R. Park, J. Kim. Biometric key binding: Fuzzy vault based on iris images. In book *Advances in Biometrics* (ed. S.-W. Lee and S. Z. Li), Heidelberg: Springer, Berlin, 2007, pp. 800–808, http://doi.org/10.1007/978-3-540-74549-5_84.

[3] Z. Akhtar, A. Hadid, M. S. Nixon, M. Tistarelli, J.-L. Dugelay, S. Marcel. Biometrics: In search of identity and security. *IEEE MultiMedia*, Vol. 25, No. 3, 2018, pp. 22–35, http://doi.org/10.1109/MMUL.2018.2873494.

[4] Smartphone market share - market share. Available Online: https://www.idc.com/promo/smartphone-market-share (visited on 29.07.2021).

[5] Mobile security project. Available Online: https://owasp.org/www- project-mobile-security/ (visited on 29.07.2021).

[6] A. Das, C. Galdi, H. Han, R. Ramachandra, J. Dugelay, A. Dantcheva. Recent Advances in Biometric Technology for Mobile Devices. *Proc. of the International Conference on Biometrics Theory, Applications and Systems*, Oct., 2018, Redondo Beach, CA, USA, pp. 1-11, http://doi.org/10.1109/BTAS.2018.8698587.

[7] M. Sandhya, M.V.N.K. Prasad. Biometric template protection: A systematic literature review of approaches and modalities. In book *Biometric Security and Privacy (ed. R. Jiang, S. Al-maadeed, A. Bouridane, P. Crookes, A. Beghdadi)*, Cham: Springer, 2017, pp. 323–370, http:// doi.org/10.1007/978-3-319-47301-7_14.

[8] A. Sarkar and B. Singh. Cryptographic key generation from cancelable fingerprint templates. *Proc. of the International Conference on Recent Advances in Information Technology*, Mar., 2018, Dhanbad, India, p. 1–6, http://doi.org/10.1109/RAIT.2018.8389007.

[9] M. Lutsenko, A. Kuznetsov, A. Kiian, O. Smirnov, T. Kuznetsova. Biometric Cryptosystems: Overview, State-of-the-Art and Perspective Directions. In book *Advances in Information and Communication Technology and Systems (ed. M. Ilchenko, L. Uryvsky, L. Globa)*, Springer, Cham, 2021, https://doi.org/10.1007/978-3-030-58359-0_5

[10] L. You, Y. Chen, B. Yan, M. Zhan. Fingerprint authentication based on fuzzy extractor in the mobile device. *International Journal of Electronic Security and Digital Forensics*, Vol. 11, No. 3, 2019, pp. 321-337, http://doi.org/ 10.1504/IJESDF.2019.100479.

[11] G. Panchal, D. Samanta. A novel approach to fingerprint biometric-based cryptographic key generation and its applications to storage security. *Computers and Electrical Engineering*, Vol. 69, 2018, pp. 461–478, http://doi.org/10.1016/j.compeleceng.2018.01.028.

[12] A. K. Trivedi, D. M. Thounaojam, S. Pal. Non-invertible cancellable fingerprint template for fingerprint biometric. *Computer Security,* Vol. 90, 2020, p. 101690, http://doi.org/10.1016/j.cose.2019.101690.

[13] S. S. Ali, I. I. Ganapathi, S. Prakash, P. Consul, S. Mahyo. Securing biometric user template using modified minutiae attributes. *Pattern Recognition Letters*, Vol. 129, 2020, pp. 263–270, http://doi.org/10.1016/j.patrec.2019.11.037.

[14] L. You, Y. Chen, B. Yan, M. Zhan. A novel location-based encryption model using fuzzy vault scheme. *Software Computing*, Vol. 22, 2018, pp. 3383–3393, https://doi.org/10.1007/s00500-017-2583-x.

[15] M.A. Alsmirat, F. Al-Alem, M. Al-Ayyoub, et al. Impact of digital fingerprint image quality on the fingerprint recognition accuracy. *Multimedia Tools Applications*, Vol. 78, 2019, pp. 3649–3688, https://doi.org/10.1007/s11042-017-5537-5.

[16] K. Nandakumar, A. Jain, S. Pankanti. Fingerprint-based fuzzy vault: Implementation and performance. *IEEE Trans. on Information Forensics and Security,* Vol. 2, No. 4, 2008, pp. 744–757, http://doi.org/10.1109/TIFS.2007.908165.

[17] Verifinger fingerprint recognition technology, algorithm and sdk for pc, smartphones and web. Available Online: https://migration.neurotechnology.com/verifinger.html (visited on 29.07.2021).

[18] M. Egele, D. Brumley, Y. Fratantonio, C. Kruegel. An empirical study of cryptographic misuse in android applications. *Proc. of the ACM Conference on Computer and Communications Security*, Nov., 2013, Berlin, Germany, pp. 73–84, http://doi.org/10.1145/2508859.2516693.

[19] H. Lipmaa, P. Rogaway, D. Wagner. Comments to NIST concerning AES modes of operation: CTR-mode encryption. Oct., 2000, Available at: https://www.cs.ucdavis.edu/~rogaway/papers/ctr.pdf (visited on 29.07.2021).

[20] V. S. Meenakshi, G. Padmavathi, Secure and revocable multibiometric templates using fuzzy vault for fingerprint and iris. In book *Information and Communication Technologies (ed. V. V. Das, R. Vijaykumar)*, Heidelberg: Springer, Berlin, 2010, pp. 206–214, https://doi.org/10.1007/978-3-642-15766-0_30.

[21] W. E. Burr, D. F. Dodson, E. M. Newton, R. A. Perlner, W. T. Polk, S. Gupta, E. A. Nabbus. Electronic authentication guideline. *National Institute of Standards and Technology*, 2013, Available at: http://dx.doi.org/10.6028/NIST.SP.800-63-2 (visited on 29.07.2021).

Information about the authors:

Sanaa Ghouzali – is an Associate Professor in the Department of Information Technology, College of Computer and Information Sciences at King Saud University (Riyadh, Saudi Arabia). Her research interests include Pattern Recognition, Biometrics, and Information Security.

Haya Altwaijri – is a Master graduate from the Department of Information Technology in King Saud University (Riyadh, Saudi Arabia). Her research interests include Information security, Android Applications Development.

Maryam Lafkih – is an Associate Professor in the Department of Computer Sciences at the Moroccan School of Engineering Sciences (EMSI), Rabat, Morocco. Her research interests include Biometrics and Information Security.

Mohammed Al-Goblan – is a Bachelor graduate from the Department of Computer Engineering in King Saud University (Riyadh, Saudi Arabia). His research interests include Signals and Systems.

Wadood Abdul – is an Associate Professor in the Department of Computer Engineering, College of Computer and Information Sciences, King Saud University (Riyadh, Saudi Arabia). His research interests are focused on multimedia security, biometrics, privacy, and medical image processing.

Manuscript received on 15 July 2021