Complex & Intelligent Systems (2022) 8:3247–3261
https://doi.org/10.1007/s40747-022-00663-3

ORIGINAL ARTICLE

Two-factor-based RSA key generation from fingerprint biometrics and password for secure communication

K. Suresh[1,2] · Rajarshi Pal[2] · S. R. Balasundaram[1]

Received: 26 August 2021 / Accepted: 22 January 2022 / Published online: 18 February 2022

Abstract
In an asymmetric-key cryptosystem, the secure storage of private keys is a challenging task. This paper proposes a novel approach for generating the same public and private key pair on a need basis. Hence, the need for secure storage of the private key is done away with. The proposed approach for generating the key pair is based on two factors: fingerprint biometrics and password. A stable binary string is generated from the distances among pairs of minutiae points in a fingerprint using a gray code-based method. Experiments show that gray code representation significantly reduces the number of inconsistencies between the generated bit strings from two instances of the same fingerprint as against the binary code representation. Hence, the Reed–Solomon error correction code successfully corrects errors due to variations in multiple instances of the same fingerprint to induce stability in the generated string. Hash of the stable string generated from the fingerprint and the string generated from hashed password are XORed to derive a stable seed value. The proposed approach uses this seed value to generate two large prime numbers. These prime numbers are used to generate the public and private key pair using the RSA key generation method. This seed value ensures the generation of the same key pair every time. The experimental results show that the proposed approach can ensure a stable generation of the key. It is not required to store either the fingerprint template or the password. Moreover, the generated private key is also not stored. It can be regenerated on a need basis.

Keywords Cryptographic Key · Bio-Cryptosystems · Fingerprint Password · Secure Communication · Reed-Solomon Code · Gray Code

Introduction

An asymmetric-key cryptosystem is mainly used for secure exchanging of session key parameters and digitally signing electronic messages. It uses a pair of keys, namely public key and private key. As these terminologies suggest, the public key is known to other entities in the ecosystem (and maybe to an intruder too). On the contrary, the private key of an entity is secret information. As a good practice, the private key is stored in separate cryptographic hardware, namely hardware security module. Though it is considered to be a safe storage for the private key, there have been attempts to steal the private key from such dedicated key storage too [1]. Hence, secure storage of the private key is also challenging. Another important point is about the randomness of the private key. There has been a report of low entropy for the space of the prime numbers being generated for RSA key derivation which leads to compromise of the key [2]. To improve the randomness, an effective combination of biometrics with cryptography, i.e., biometric cryptosystem [3,4] has emerged.

In a biometric cryptosystem, biometrics is used either to access the cryptographic key (key release) [5,6], to retrieve the cryptographic key without explicitly storing it (key binding) [7–11] or to generate a cryptographic key directly from the biometric features (key generation) [12,13]. Secure transmission/storage and authentication are performed based on the key, which is either accessed, retrieved, or generated using the biometric template.

✉ K. Suresh
suresh.1088@gmail.com

Rajarshi Pal
iarajarshi@yahoo.co.in

S. R. Balasundaram
blsundar@nitt.edu

1 Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Tamil Nadu, India

2 Centre of Excellence in Cyber Security, Institute for Development and Research in Banking Technology, Hyderabad, Telangana, India

In this work, two-factor-based RSA cryptographic key pair generation is proposed. Reed–Solomon error correction code is applied to correct the mismatch bits to provide stability to the regenerated key string from fingerprint biometrics. A standard password-based key derivation function is incorporated to strengthen the security of the generated key pair and, hence, improves the robustness of the proposed method.

The highlights of this work are given as follows:

1. Two-factor fingerprint biometrics and password are used to generate the public and private key pair which strengthens the security of the key pair.
2. A gray code-based encoding is used to generate a stable key string from fingerprint biometrics. Reed–Solomon error correction code is used to correct the errors arising due to biometric variance to provide stability to the generated key string.
3. A standard password-based key derivation function is used to generate a stable string from a user-specified password.
4. According to the proposed scheme, the biometric information, password and the generated key pair is not stored anywhere. Therefore, blended substitution attack, key inversion attack and attack via record multiplicity are not possible.
5. This approach generates two distinct cryptographic key strings for two different users. Therefore, the chance of producing the same cryptographic key string by a genuine user and an impostor is eradicated.
6. The various security analysis methods ensure the security of the generated cryptographic key pair.

The rest of the paper is organized as follows. A brief overview of existing key generation approaches from biometrics is described in the next section. The proposed two-factor-based stable RSA cryptographic key pair generation process is deliberated in the third section Experimental results are reported in the fourth section. In the fifth section, the security analysis of the proposed work is presented. Finally, the contributions of this paper are concluded in the sixth section.

Cryptographic key generation from biometrics: literature survey

In key generation scheme, a cryptographic key is generated from extracted unique features of the user's biometric traits. This paper deals with a key generation scheme. Hence, the subsequent discussion highlights the available methods of cryptographic key generation from biometrics. An overview of generating cryptographic keys from various biometric traits has been shown in Fig. 1.

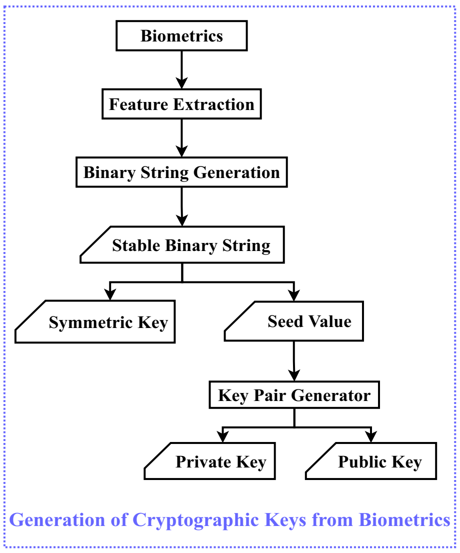


Fig. 1 Generation of cryptographic keys from biometrics

Approaches for generating cryptographic key from biometrics have considered various biometric traits, such as fingerprint [12–17] finger vein [18], iris [19], face [20,21], voice [22], handwritten signature [23], gait [24–27], ECG [28,29], retina [30] and multi-modal [31–33].

Fingerprint is the most widely used biometric trait among all these traits for practical applications. Hence, in the current paper, fingerprint biometrics is considered as a part of cryptographic key pair generation. Here, several existing approaches of stable binary string generations are briefly discussed. The generated stable binary string can be used directly as a cryptographic key for symmetric encipherment [12–15] or it can be used as a seed value to generate the private–public key pairs for asymmetric encipherment [34,35].

In [12], a cryptographic key generation technique is proposed using fingerprint biometrics. The fingerprint biometric is partitioned into a set of non-overlapping blocks. Every pair of minutiae points from two neighboring blocks is joined using straight line. The binary representations of orientation angle and length of each straight line are XORed. Then, these XORed strings of all straight lines are concatenated. A sequence of substitution, expansion, and permutation of the concatenated string generates a stable binary string. This stable binary string is used as a symmetric cryptographic key. The length and the angle values of these straight lines are used to generate helper data. Similarly, another approach of generating a key is proposed in [14] from length and orientation of straight lines. In [15], distances between every pair of minutiae points are estimated. Subsequently, unique distances are sorted in ascending order. A binary string is generated whose length equals the maximum distance value. If the index value of an element in the string is present in the sorted sequence of distances, then the bit in the string at the index position is set as 1. If the index value is not present in the sorted sequence of distances, then the concerned bit is 0. This binary string acts as the cryptographic key. A similar strategy is also adopted in [16]. However, a small variation in [16] is that a random permutation of the binary string is used as the cryptographic key. The permuted bit positions of the binary string are stored as helper data.

In [17], feature distance set is generated by calculating distances between every pair of minutiae points. A generated intervals are constructed from the feature distance set. The generated intervals are encoded to obtain the bio-key. The bio-key is encoded using two-layer error correction code (Hadamard and Reed–Solomon) to identify and correct errors. The generated intervals, auxiliary data (differences of minutiae types and orientation fields), error correction codeword and hash value of bio-key are stored as helper data in a smart card. During key regeneration phase, these stored helper data are used to regenerate the bio-key from the query fingerprint image. The hamming distance between the hash value of the stored bio-key and the hash value of regenerated bio-key is computed to verify the regeneration process. If the hamming distance is within the predefined threshold the regeneration is successful otherwise failed.

In [38], Euclidean distances are calculated between each pair of minutiae points. The sequence of minutiae points is obtained by sorting the euclidean distances in ascending order. A Trellis diagram is generated from this minutiae point sequence using a convolution coder, which leads to a unique code. Further, a hash of the unique code produces the cryptographic key.

In most of the above methods, helper data are used to regenerate the cryptographic key. However, helper data-based cryptographic key generation methods are vulnerable to masquerade attack [36], brute force attack [36], attack via record multiplicity [37] and attack on host [35]. Moreover, reconstruction of the original biometrics may be possible from the stored helper data [37]. In all the above methods, a stable binary string is generated from the extracted features of the biometric traits, which is further used as a cryptographic key for symmetric key encipherment.

In the present era, researchers are working on generating asymmetric key pairs from biometric traits. Very few works has been proposed a framework for generating private–public key pairs for secure exchanging of session key parameters and digitally signing electronic messages. In [34], the authors proposed a technique to generate a pair of private and public key for RSA cryptosystem using fingerprint biometrics. Minutiae points are extracted from the fingerprint biometrics and generated a cancellable template by shuffling the x and y coordinate values and concatenated by performing bitwise XOR operation among the x and y coordinates of each minutiae point. The cancellable template is salted to extract the desired key length bits for the RSA cryptosystem. The salted cancellable is used to generate the two large prime numbers, which is used to generate the private–public key pair. The x and y coordinate values are directly used for the key generation process. Due to biometric uncertainty, this method does not ensure the stability of the key.

In [35], the authors proposed a technique to generate a common secret key from the fingerprints of sender and receiver using public key cryptography. Each person generates a binary string from the extracted features of their own fingerprint. User specific transformation key is used to generate a random numbers equal to the length of the feature binary string. Each person performs a permutation on their generated binary string using this random numbers. Then the permuted binary string is hashed with SHA-256 algorithm to generate a 256-bit private key which is used as an input to Diffie–Hellman key exchange algorithm along with two predefined parameters to generate public keys of sender and receiver. The generated public keys are shared between the sender and receiver. Then, Diffie–Hellman algorithm uses user's own private and other user's public key to generate a common secret key among both the users. This common secret key is hashed with SHA-256 algorithm to generate final symmetric cryptographic key.

In [39], an approach is proposed for generating a cryptographic key using the shared minutiae points from fingerprints of the communicating parties at both ends. Each communicating party creates a cancellable template from his fingerprint using a shuffling-based minutiae point transformation scheme and bitwise XOR operation between x and y coordinates of each minutiae point. Then, the sender and receiver shared their generated cancellable templates. A master template is generated using the templates of both communicating parties. Finally, the master template is used to derive the key by considering equal contribution points from both communicating parties templates. Similar techniques are used in [40–44] to generate the cryptographic key from fingerprint biometrics of both communicating parties.

In the majority of the above approaches, cryptographic key (or part of it) is obtained by the binary encoding of decimal values (e.g., principal component [20], length and angle ratio [14], the Euclidean distances [15,16]). In binary code-based approaches, small variations in these values have the maximum effects on the generated key bit strings. As a result, more mismatch bits need to be corrected using error correction code. Contrary to this binary encoding, gray code-based encoding is presented in [13]. The use of gray code significantly reduces the number of mismatching bits between the key strings generated from two different instances of the same fingerprint.

In all these above methods, the stable binary string or the seed value is directly generated from biometric traits. But the compromise of biometrics [36] leads to the weakness of

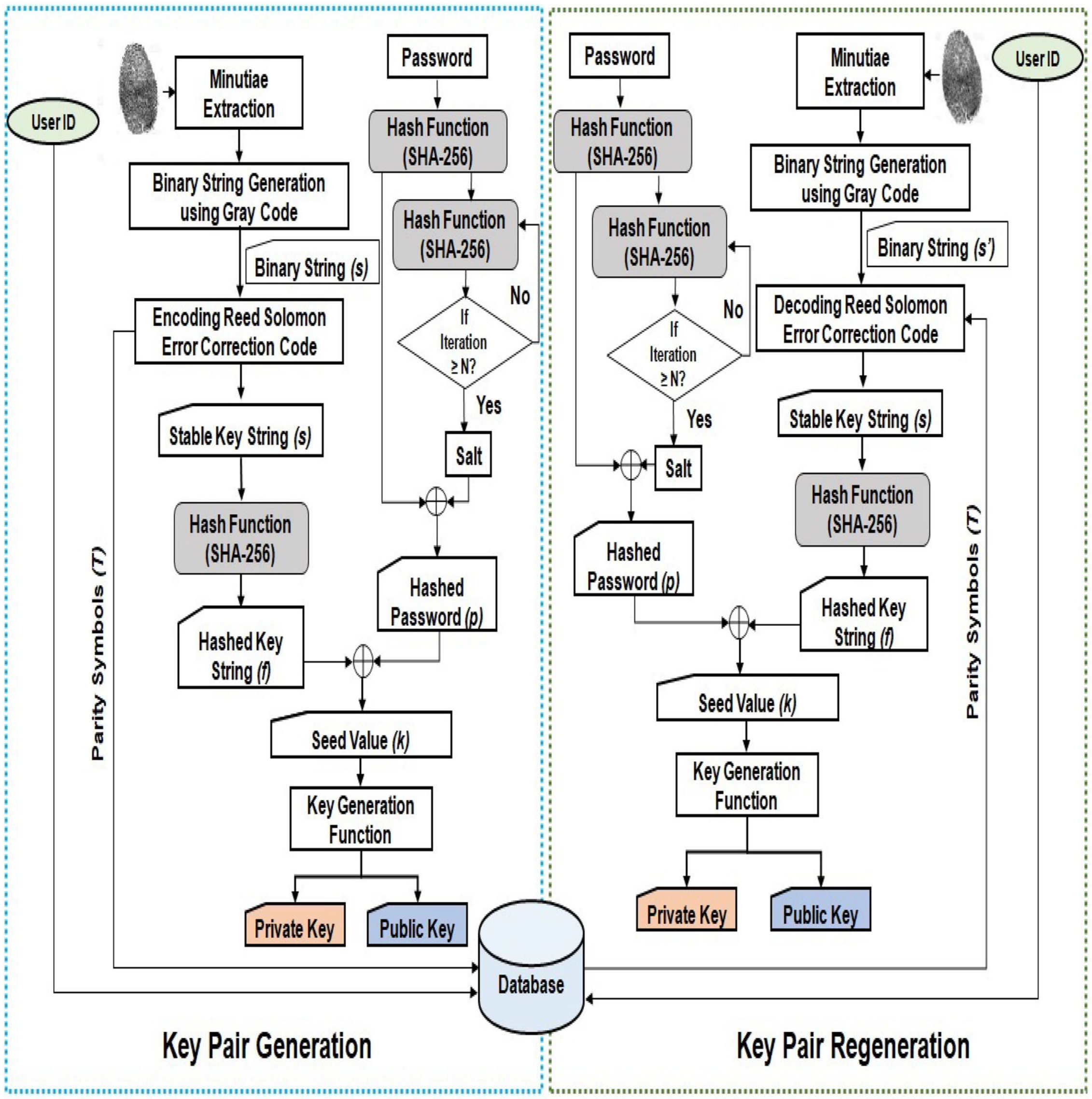


Fig. 2 Proposed two-factor-based RSA cryptographic key pair generation from fingerprint biometrics and password

the above methods. To overcome this weakness, two-factor-based cryptographic key generation is proposed in this paper. The first key string is generated from the user's fingerprint biometrics, and the second key string is generated from a secret password of the user. Both the key strings are XORed to generate the stable seed value. Hence, compromise of any one factor (either fingerprint or password) will not lead to the compromise of the generated key pair. Usage of two factors distinguishes the proposed work from all other existing works (including work in [13]). Moreover, similar to the work in [13], the current work also adopts a gray code encoding of decimal values to generate a key bit string from a fingerprint image. But unlike the work in [13], another distinguishing feature of the proposed work is the use of error correction code to handle variations due to the intrinsic uncertainty of acquiring fingerprint biometrics. In this work, the Reed–Solomon error correction code is applied to correct the mismatching bits to stabilize the regenerated key string. This is an essential step, as precisely the same key bit string needs to be regenerated every time to use it to generate the same public and private key pair. In addition, hash function is used to increase the randomness and resistance to brute force attacks.

Proposed method

In this paper, two-factor-based stable RSA cryptographic key pair generation is proposed using fingerprint biometrics and password. An overview of generating a stable cryptographic key pair from fingerprint biometrics and password is diagrammatically shown in Fig. 2. The detailed steps for the proposed scheme are discussed in this section.

Stable string generation from fingerprint

The steps involved in stable string generation from fingerprint biometrics are shown in Algorithm 1.

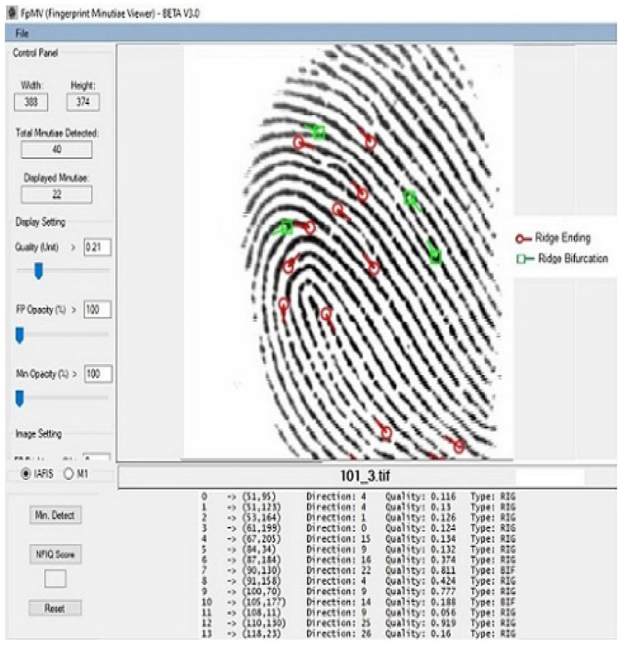
Algorithm 1 Generation of Stable String from Fingerprint Biometrics

```
1: Input: Fingerprint Image
2: Detect minutiae points using contrast enhancement. /* To improve
   the quality of the fingerprint image */.
3: Apply binarization. /* To convert the fingerprint gray scale image
   into a binary image */.
4: Apply thinning. /* To reduce the widths of the binarized image ridge
   lines into 1-pixel */.
5: Select consistent set of minutiae points $m_1, m_2, .. m_i$.
6: Calculate Euclidean distances ED. /* Distance between each pair of
   minutiae points */.
7: Sort Euclidean distances ED. $S_{E\ D}$ = Sort ($ED_1, ED_2, ..., ED_n$). /*
   Ascending order */
8: Convert $S_{E\ D}$ to 8- bit binary number $B_1, B_2, \dots, B_n$.
9: Convert $B_1, B_2, \dots, B_n$ into gray code $G_1, G_2, \dots, G_n$.
10: Concatenate gray code $G = G_1 \cup G_2 \cup \dots \cup G_n$
11: Apply SHA-256 algorithm over $G$ to generate stable bit string $f$
    from fingerprint.
12: Output: 256-bit Stable String $f$ from fingerprint
```

Minutiae extraction: Minutiae points in a fingerprint uniquely characterize each individual. Hence, accurate minutiae points from a captured fingerprint image are necessary to generate a stable key string for each individual. Ridge bifurcation and ridge ending are the most widely accepted minutiae point [45]. Hence, only ridge ending and ridge bifurcation points are used in this proposed work. A point where a ridge ends or terminates immediately, is termed as a ridge ending. A point where two ridge lines are obtained by splitting a single ridge line is termed as ridge bifurcation.

In the proposed approach, similar to [39,41,43,44], NIST's NBIS software tool Fingerprint Minutiae Viewer (FpMV) [46] is used for extracting the minutiae points. This tool gives the output as minutiae coordinate values (x, y), type of minutiae, quality, and angle. The true and good quality minutiae points are extracted by adjusting the value of the quality parameter in the FpMV tool for each fingerprint image. A sample snapshot is shown in Fig. 3.

Due to biometric uncertainty in capturing the same biometric signal, two acquired images of same fingerprint may not be always same. Hence, the acquired images do not have exactly the same set of minutiae points. Obtaining a stable key string from fingerprint biometrics is the main objective of this proposed work. Hence, a consistent set of minutiae points must be used for key string generation. In this work, a



Width: 388 Height: 374
Total Minutiae Detected: 40
Displayed Minutiae: 22
Quality (Unit) > 0.21
FP Opacity (%) > 100
Min Opacity (%) > 100
[✓ IAFIS] M1

101_3.tif

```
0    -> (51,95)     Direction: 4    Quality: 0.116   Type: RIG
1    -> (51,123)    Direction: 4    Quality: 0.13    Type: RIG
2    -> (53,164)    Direction: 1    Quality: 0.126   Type: RIG
3    -> (61,199)    Direction: 0    Quality: 0.124   Type: RIG
4    -> (67,205)    Direction: 15   Quality: 0.134   Type: RIG
5    -> (84,34)     Direction: 9    Quality: 0.132   Type: RIG
6    -> (87,184)    Direction: 16   Quality: 0.374   Type: RIG
7    -> (90,130)    Direction: 22   Quality: 0.811   Type: BIF
8    -> (91,158)    Direction: 4    Quality: 0.424   Type: RIG
9    -> (100,70)    Direction: 9    Quality: 0.777   Type: RIG
10   -> (105,177)   Direction: 14   Quality: 0.188   Type: BIF
11   -> (108,11)    Direction: 9    Quality: 0.056   Type: RIG
12   -> (110,130)   Direction: 25   Quality: 0.919   Type: RIG
13   -> (118,23)    Direction: 26   Quality: 0.16    Type: RIG
```

Fig. 3 Minutiae extraction using Fingerprint Minutiae Viewer tool

consistent set of minutiae points is selected to generate a stable key string from the common region of all the fingerprint images of the same subject using NIST's NBIS software tool Fingerprint Minutiae Viewer (FpMV) [46].

Binary string generation using gray code Euclidean distance between each pair of minutiae points is calculated from the selected consistent set of minutiae points. Let two minutiae points P_i and P_j be located in the coordinates (x_i, y_i) and (x_j, y_j), respectively. The Euclidean distance between this pair of minutiae points is computed as

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{1}$$

Similar to several other works [15,16,38,47], Euclidean distance between pair of minutiae points is considered here due to its rotational and translational invariance. In spite of rotation- and translation-induced changes in the absolute coordinate values of the minutiae points, the distance between each pair of minutiae points does not vary. Hence, it does not affect the stable key string generation process.

If N minutiae points are finally selected from a fingerprint, ${}^{N}C_2$ such distances are calculated from the coordinate values of minutiae points. These ${}^{N}C_2$ Euclidean distance values are sorted in ascending order to generate the consistent key bit string.

Each Euclidean distance in the sorted sequence is converted into a c-bit binary number. Then, the c-bit binary number is converted into c-bit gray code. Gray code is a reflected binary code in which two successive values differ in only one bit. Gray code provides a major advantage to this proposed work. Due to biometric uncertainty, if there is slight change in the distance value, then there will be only minimum change on the generated bit string. Therefore, it can be easily corrected using error correction codes.

For example, Euclidean distance values of 63 and 64 are represented as ‘00111111’ and ‘01000000’, respectively, as 8-bit binary form. Even there is a small change in distance by a value of 1 from 63 to 64, the number of mismatching bit positions in their corresponding binary strings is 7. On the contrary, the values of 63 and 64 are represented as ‘00100000’ and ‘01100000’, respectively, 8-bit gray code. These strings vary in only one bit position. Therefore, the use of gray code significantly reduces the number of mismatching bits which may arise due to slight variation in the Euclidean distances.

Let a c-bit binary string be ‘$b_0 b_1 b_2 b_3 ... b_{c-1}$’. Let the corresponding gray code be ‘$g_0 g_1 g_2 g_3 ... g_{c-1}$’. The conversion from binary to gray code is carried out as following:

$$g_0 = b_0$$
$$g_i = b_{i-1} \oplus b_i,$$

for i = 1 to $c - 1$ where the symbol ’$\oplus$’ represents bitwise exclusive-OR operation.

The c-bit representations of each gray code are concatenated to attain a single key bit string. For the selected N minutiae points, there exist ${}^N C_2$ distance values. Hence, the length of key bit string is $c \times {}^N C_2$ bits.

Generation of parity symbols using Reed–Solomon code Due to biometric uncertainty, two acquired images of same fingerprint may not be exactly the same. Hence, two slightly different binary key strings s and s' may be derived by following the proposed key string generation process. But to generate a same key pair, the exactly same seed value is required each time. Let the generated binary key be s during the key generation (Fig. 1). Then, the regenerated binary key string s' (during regeneration of the same key) is treated as erroneous. Reed–Solomon error correction code [48] is used, in this work, to correct the erroneous bits in the regenerated binary key s'. Reed–Solomon code is able to detect and to correct multiple or burst symbol errors. By adding t ($t = n - m$) check symbols to the generated gray code data, a (n, m) Reed–Solomon code can detect mismatch up to t erroneous symbols. It can also locate and correct the errors up to $\lfloor t/2 \rfloor$ erroneous symbols at any locations. The symbol $\lfloor \ \rfloor$ indicates the greatest integer having value less than its argument. Reed–Solomon code commonly uses a finite field F with 2^l elements. Each symbol is represented as an l-bit value. In this current work, the value of l is considered to be 8. If m bits of binary string s is encoded with a capacity of correcting up to t bits of error, Reed–Solomon code generates m 8-bit

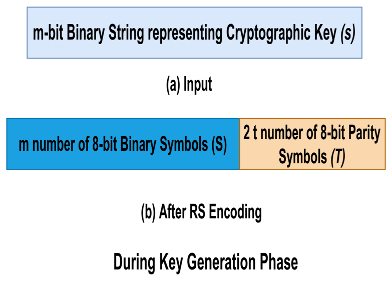

m-bit Binary String representing Cryptographic Key (s)
(a) Input
m number of 8-bit Binary Symbols (S)
2 t number of 8-bit Parity Symbols (T)
(b) After RS Encoding
During Key Generation Phase

Fig. 4 Generation of parity symbols using Reed–Solomon code

data symbols (S) and $2t$ 8-bit parity symbols (T). The generated sequence of parity symbols T is stored in the database. During key string regeneration process, this sequence of parity symbols T is used to correct the errors which may arise due to biometric uncertainty. The generation of parity symbols using Reed–Solomon code is shown in Fig. 4.

Hashed key string generation: The length of the stable bit string s is $c \times {}^N C_2$ bits. To generate the key string having a consistent length of 256 bits, the generated stable bit string s is hashed with the SHA-256 algorithm. This hash value f is considered as the key string from fingerprint biometrics.

Password-based string generation

The steps involved for generating stable string from password is deliberated in Algorithm 2.

Algorithm 2 Generation of Stable String from Password

```
1: Input: Password (P)
2: Convert password P into a binary string B. /* by concatenating 8-bit
   ASCII representation of symbols in the password (P) */.
3: Apply SHA-256 algorithm over B (i.e., h(B)).
4: Generate salt value S.
5: Generate 256-bit $p = h(B) \oplus S$, '$\oplus$' represents bit-wise exclusive-
   OR operation.
6: Output: 256-bit Stable String $p$ from password
```

Unlike the work in [13], another distinguishing feature of the proposed work is the usage of two-factor-based key generation. Apart from fingerprint biometrics, password is also used to generate a key pair to enhance the security of the proposed work. NIST recommends a technique for deriving a cryptographic key from password to protect stored electronic data [49]. Based on NIST recommendation, existing methods [50,51] use password-based key derivation functions (PBKDF) by specifying a fixed iteration count and a pseudo-random function. The inputs to a PBKDF include a password, a salt and a desired key length [50,51]. Similarly, in this proposed work, the key is derived using a hash computation-based approach. But unlike the approaches in [50,51], the salt is derived from the user-specified password.

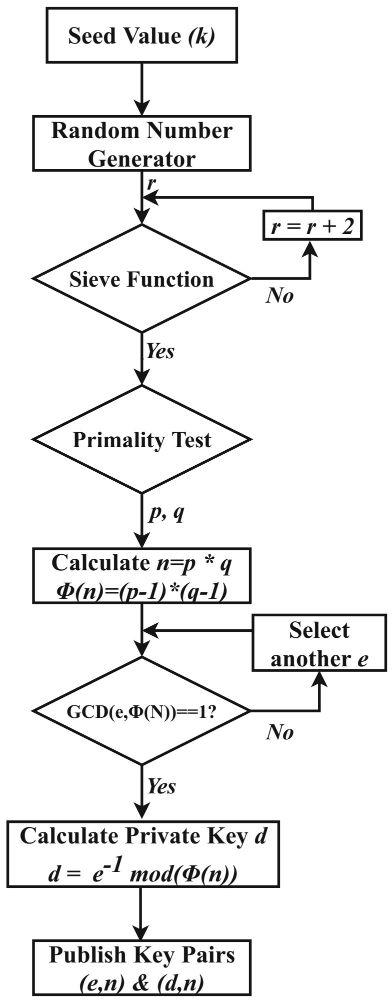


Fig. 5 Generation of RSA key pair from seed value

Hence, there is no need for the user to remember both of password and salt(or any other parameter to derive the salt). Moreover, composition of the user-specified password must follow the NIST guidelines [52]. The important clauses about the password formation are:

1. The length of the password should be at least eight symbols.
2. The password comprises a mixture of upper case and lower case alphabets, numbers and special symbols.
3. It does not contain any dictionary word, personal information, series of characters or repeated characters.

The approach for key derivation from a password is detailed here. The password is converted into a binary string by concatenating 8-bit ASCII representations of the symbols in the password. Hash of this binary string is computed using SHA-256 algorithm. Similar to the works in [50,51], a salt is used to decouple the binary string from the password. It will prevent brute force and dictionary attacks. In the proposed method, the salt is generated from the previous hash value of the binary password string. The hashed binary password is further hashed iteratively for n times using the SHA-256 algorithm to generate a salt value. In this work, 10,000 iterations of the hash computation are carried out to generate the salt value. The large number of iterations makes it impractical for the attacker to implement rainbow table attacks. The hashed binary password and the generated salt value are then XORed to generate the 256-bit key string p from password. This process of deriving the binary key from password is demonstrated as part of Fig. 1.

Generation of stable seed value

The generated key string f from the fingerprint biometrics and the password key string p are XORed to generate a stable seed value k. Stability in the key f from the fingerprint through Reed–Solomon Error correction code ensures the stability of the seed value k.

Algorithm 3 Generation of Stable Seed value from Fingerprint image and Password

```
1: Input: 256-bit stable string $f$ from fingerprint, 256-bit string $p$
   from password
2: $k = f \oplus p$, '$\oplus$' represents bit-wise exclusive-OR operation.
3: Output: 256-bit Stable Seed Value $k$
```

Generation of RSA key pair

The generation of public and private key pair from seed value is shown in Fig. 5.

Generation of large prime numbers

A pseudo-random number generator (PRNG) is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers. The PRNG-generated sequence is not truly random, because it is completely determined by an initial value, called the PRNG's seed. Here, in this work, the generated stable seed value k is used as input to random number generator. After generating a random number, its low-order bit is set to 1 which ensures that the generated number is odd.

Sieve function

Before performing the primality test, the generated random numbers are fed into the sieve function to detect composite numbers by a trail modular computation method [53]. Since the procedure of primality test is more time consuming, to decrease the number of test iterations sieve function unit is used before primality test.

Primality test

To check whether the generated random number is prime or not the primality test function is used. There are two distinct methods namely: deterministic and probabilistic. In the deterministic test, Fermat's theorem [54] is used to specify the primality of Sieve survivor. It states that if p is a prime number, for any integer a relatively prime to p that $1 \leq$ a $< p$, then $a^{p-1} mod p = 1$. The operation of the primality test unit is based on extensive modular exponentiation operation. The probabilistic tests use Miller Rabin, Fermat, and Solovay–Strassen tests [54,55] to determine the primality of a number with a given degree of confidence.

Key generation process

The steps for generating N-bit key pair are as follows:

1. Generation of two distinct $(N/2)$-bit prime numbers p and q using seed value k.
2. Calculate $n = p * q$.
3. Calculate $\phi(n) = (p-1)*(q-1)$, where n is the modulus function and $\phi(.)$ is the Euler's phi function [].
4. Calculate the Public key e, which satisfies $1 < e < \phi(n)$ and e is relative prime to $\phi(n)$ (i.e., $GCD(e, \phi(n)) == 1$), where GCD is Greatest Common Divisor).
5. Calculate the Private key d, which is modular inverse of e modulo $\phi(n)$ (i.e., $d \equiv e^{-1}(mod \phi(n))$).

Encryption

The generated public key $\{e, n\}$ is sent to the receiver. The receiver encrypts the sensitive personal data(L) using the received pair $\{e, n\}$ to generate the cipher text C as $C = L^e \ mod \ n$. The generated cipher text C is transmitted over insecure channel to the sender.

Decryption of encrypted data

Finally, the sender decrypts the received cipher text C using his private key $\{d, n\}$ to retrieve the sensitive personal data (L) as $L = C^{\ d} \ mod \ n$.

Regeneration of same key pair

The query fingerprint image and password are acquired from the user to regenerate the same cryptographic key pair on need basis. User ID is used to retrieve the stored parity symbols T data from the database. The major steps involved for regeneration of same key pair are as follows:

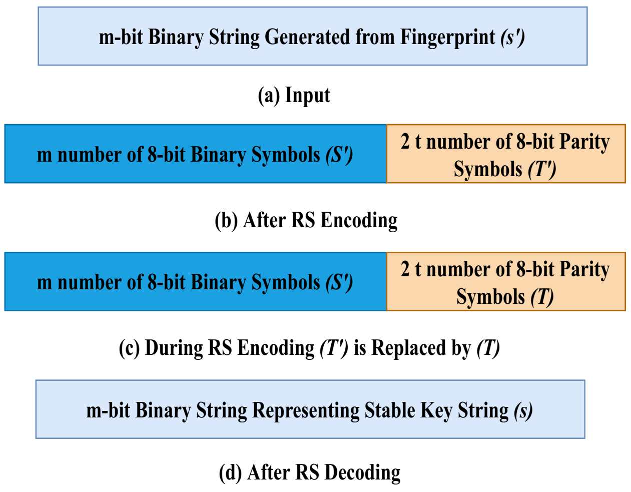

m-bit Binary String Generated from Fingerprint (s')
(a) Input
m number of 8-bit Binary Symbols (S')
2 t number of 8-bit Parity Symbols (T')
(b) After RS Encoding
m number of 8-bit Binary Symbols (S')
2 t number of 8-bit Parity Symbols (T)
(c) During RS Encoding (T') is Replaced by (T)
m-bit Binary String Representing Stable Key String (s)
(d) After RS Decoding

Fig. 6 Regeneration of stable key string using Reed–Solomon code

Regeneration of stable binary string from fingerprint using Reed–Solomon code

The seed value must be same for generating the same key pair every time. Hence, there is a necessity to correct the errors which arose due to biometric variance. It is to be noted in Fig. 1 that a key string s was derived during the key pair generation (Sects. 3.1 and 3.2). Similar steps are followed to generate a binary key from fingerprint during key pair regeneration process too. But during regeneration phase, a slightly different key string s' is generated due to biometric variance. The following steps regenerate the original key string s from s' using the stored parity symbols T:

1. The m bits of generated binary string s' from the query fingerprint image is encoded with Reed–Solomon error correction code having a capacity of correcting up to t number of bits of error. It generates m number of 8-bit data symbols (S') and $2t$ number of 8-bit parity symbols (T').
2. The generated sequence of parity symbols T' is replaced with the stored sequence of parity symbols T.
3. During decoding, the stored parity symbols T correct the errors which arose due to biometric variance and regenerate m-bit binary key string s.

The regeneration of stable key string s using Reed–Solomon error correction code is shown in Fig. 6.

Regeneration of stable seed value using two factors

The regenerated key string s from the presented fingerprint biometrics is hashed using SHA-256 algorithm to generate the key string f from query fingerprint. Similar to step 3.1.1, a

Table 1 Fingerprint verification competition (FVC) 2002 dataset

DB No	Sensor	Number of images	Number of persons	Number of samples	Image size	Resolution (dpi)
DB1 (Set B)	Optical sensor "TouchView II" by Identix	80	10	8	388 × 374 (142 K pixels)	500
DB2 (Set B)	Optical sensor "FX2000" by Biometrika	80	10	8	296 × 560 (162 K pixels)	569
DB3 (Set B)	Capacitive sensor "100 SC" by Precise Biometrics	80	10	8	300 × 300 (88 K pixels)	500

key string p is generated from password. The two key strings f and p are XORed to generate the stable seed value k.

Generation of RSA key pair

Further, similar to steps in 3.2, the stable seed value k is used to generate the same public and private key pair.

Experimental results and analysis

In this section, experimental results are presented to show the usefulness of gray code in generating a key string s from the fingerprint.

Establishing the usefulness of gray code for key string generation from fingerprint

In this work, experiments are reported using Fingerprint Verification Competition (FVC) 2002 Dataset DB1, DB2 and DB3 (set B of each of three datasets) [56]. There is a total of 80 (10×8) fingerprint images in each dataset. Each of these datasets has fingerprints from 10 subjects with 8 fingerprint images of each subject. In DB1, the size of each fingerprint image is 388 × 374, which were acquired at 500 dpi (dots per inch). In DB2, the size of each fingerprint image is 296 × 560, which were acquired at 569 dpi. In DB3, the size of each fingerprint image is 300 × 300, which were acquired at 500 dpi. The details of the datasets are shown in Table 1.

In this work, similar to [39,41,43,44], NIST's NBIS software tool Fingerprint Minutiae Viewer (FpMV) [46] is used for extracting the minutiae points. In this work, coordinate values of ridge ending and ridge bifurcation minutiae points are used to generate the key strings. Few additional implementational details are discussed here: The point- to -point correspondences among extracted minutiae points from various fingerprint images of the same person are used to manually select a consistent set of 7 minutiae points from each fingerprint. As a result, $^7C_2$=21 distances are obtained. The measured Euclidean distances for the DB1 and DB3 datasets are observed to be in the range [0, 255]. Hence, 8-bit gray code representation is used to encode the distances among minutiae points for DB1 and DB3. Therefore, the length of the obtained key for DB1 and DB3 is: 21 × 8 = 168 bits. The distances in the DB2 datasets are observed in the range of [0, 300]. Hence, 9-bit gray code representation is used to encode each distance between minutiae points for DB2. Therefore, length of the obtained key from fingerprint image in DB2 is: 21 × 9 = 189 bits.

Due to biometric uncertainty, there will be slight mismatches between the generated key strings from two acquired images of same fingerprint. These mismatches can be corrected using error correction codes, such as Reed–Solomon code [10] and Hadamard code [57]. In this work, the mismatches among the generated key strings are analyzed to estimate the feasibility of the proposed scheme before actually applying the error correction code. This analysis is carried out to find the maximum number of bits which need to be corrected by the error correction code. Hamming distance is used to estimate the difference between two fingerprint image key strings (in terms of the number of mismatched bits).

In each dataset, there are a total of 80 fingerprint images (8 fingerprint images for each of 10 subjects). Hence, $^8C_2 = 28$ Hamming distances can be obtained between the generated key strings of each pair of fingerprints of same subject. Therefore, there are maximum 280 (i.e., 28 × 10) such possible Hamming distances. In addition, the dissimilarity between the generated key strings of a genuine user and an impostor are assessed to check the feasibility of this proposed work. Hence, Hamming distance is calculated between a pair of genuine and impostor key strings. Each genuine user's fingerprint is compared to each impostor's fingerprint in the dataset (i.e., 8 fingerprints × 9 impostors = 72 comparisons for each fingerprint). As a result, 5760 (72 comparisons/fingerprint × 80 fingerprints) Hamming distance estimations are possible in total for impostor category.

In DB1 dataset, 12 fingerprint images out of 80 images are observed to have either partial fingerprints or low quality. Hence, it is not possible to extract the desired minutiae points from those 12 fingerprint images. Therefore, 205 key pairs are compared to estimate the Hamming distances between the

Table 2 Simple statistics of Hamming distances between pairs of generated key strings of gray code and binary code for genuine and impostor categories for DB1, DB2 and DB3 datasets

Code	Category	DB1			DB2			DB3		
		Minimum	Average	Maximum	Minimum	Average	Maximum	Minimum	Average	Maximum
Gray code	Genuine	4	12.46	20	3	11.16	19	3	10.92	18
	Impostor	40	64.63	81	42	67.10	96	40	66.81	93
Binary code	Genuine	9	24.05	42	5	21.80	41	8	22.50	40
	Impostor	36	68.18	96	39	75.07	97	36	70.96	98

key strings being derived from two fingerprint instances of the same subject (i.e., genuine case). The observed maximum Hamming distance is 20 and the average Hamming distance is 12.46. Similarly, 4146 Hamming distances are estimated for impostor category. In the impostor category, the minimum Hamming distance is 40 and the average Hamming distance is 64.63. These values are reported in Table 2.

Similar experiments are carried out using DB2 and DB3 datasets. In DB2 dataset, 8 fingerprint images out of 80 images are observed to have either partial fingerprints or low quality images. Hence, it is not possible to extract the desired minutiae points from those 8 fingerprint images. Therefore, 224 key pairs are compared to estimate the Hamming distances in the genuine category. The observed maximum Hamming distance is 19 and the average Hamming distance is 11.16. Similarly for impostor category, a total of 4664 Hamming distances are estimated. In the impostor category, the observed minimum Hamming distance is 42 and the average Hamming distance is 67.10. These values are reported in Table 2.

In DB3 dataset, 9 fingerprint images out of 80 images are observed to have either partial fingerprints or low quality. Hence, it is not possible to extract the desired minutiae points from those 9 fingerprint images. Hence, 219 key pairs are compared to estimate the Hamming distances in the genuine category. The observed maximum Hamming distance is 18 and the average Hamming distance is 10.92. Similarly, for the impostor category, 4532 key pairs are compared. The observed minimum Hamming distance is 40 and the average Hamming distance is 66.81. These values are reported in Table 2. From the reported values in Table 2 corresponding to usage of gray code in generating the key string from fingerprint, it can be seen that the maximum Hamming distance for genuine category is smaller than the minimum Hamming distance for impostor category for each of the three datasets. Hence, it can be concluded that an error correction code with suitable parameter values will be able to correct the mismatching between the pair of key strings in a genuine case. But it will not correct all the mismatching bits in an impostor case.

To demonstrate the effectiveness of using gray code to represent the Euclidean distances between minutiae points, key strings are also generated using standard binary encoding of Euclidean distances instead of gray code encoding. Binary code-based key strings are compared using Hamming distance for each of 205 pairs of key strings of genuine users for DB1. The observed maximum Hamming distance is 42 and the average Hamming distance is 24.05. Similarly, binary code-based key strings are also compared for 4146 pairs of key strings in impostor cases using Hamming distance. The observed minimum Hamming distance is 36 and the average Hamming distance is 68.18. These values are reported in Table 2.

Similar comparisons with the generated key strings using binary code representation are also performed for DB2 and DB3 datasets. In DB2 dataset, for genuine case, 224 Hamming distances of pairs of generated key strings are computed. The observed maximum Hamming distance is 41 and the average Hamming distance is 21.80. Similarly for impostor cases, 4664 key pairs are compared. The observed minimum Hamming distance is 39 and the average Hamming distance is 75.07. These values are reported in Table 2.

In DB3 dataset, for genuine cases, 219 Hamming distances between pairs of generated key strings are computed. The observed maximum Hamming distance is 40 and the average Hamming distance is 22.50. Similarly, for impostor cases, 4532 key pairs are compared. The observed minimum Hamming distance is 36 and the average Hamming distance is 70.96. These values are reported in Table 2. It can be observed from these reported values that the minimum Hamming distance for an impostor case is smaller than the maximum Hamming distance in genuine case across all three datasets. It is clear from the experiments that the binary code-based key from an impostor fingerprint may match with the binary code-based key from a genuine fingerprint.

Distributions of Hamming distances between pairs of gray code and binary code-based key string for genuine and impostor fingerprints are presented in Fig. 7 for DB1, DB2 and DB3 datasets. There is overlapping between the distributions of Hamming distances for genuine and impostor scenarios. On the contrary, in the gray code-based approach of generating a key string, these distributions of Hamming distances are well separated for genuine and impostor categories are shown in Fig. 7. Therefore, it is impossible for an impostor to gen-

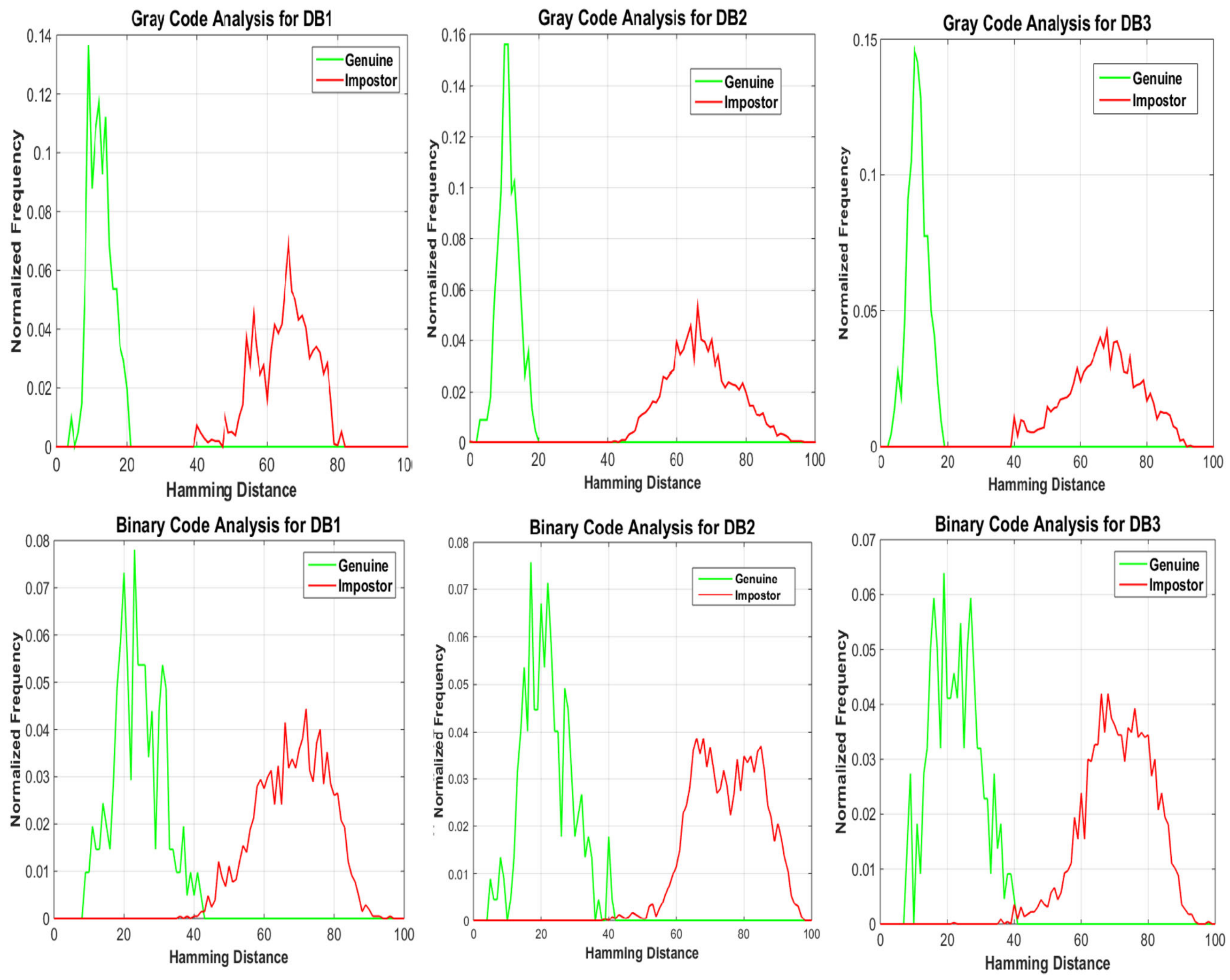


Fig. 7 Distributions of Hamming distances between pairs of gray code and binary code-based generated key strings for genuine and impostor fingerprints for DB1, DB2, DB3

Table 3 The parameters for Reed–Solomon code for three datasets DB1, DB2, and DB3

Dataset	Length of generated bit string from fingerprint	Reed–Solomon code parameter (n, m)	Error correction capability (t)
DB1	168	(208, 168)	20
DB2	189	(229, 189)	20
DB3	168	(208, 168)	20

erate the same key string as a genuine user. The generated key string from the impostor's fingerprint is significantly different from the key string generated from a genuine user's fingerprint. Hence, an error correction code should be able to successfully correct the mismatching bits to generate a stable key from genuine fingerprint. The parameters of the error correction code must be selected such that it will not be able to correct the bits in the impostor cases. Hence, in this proposed work, Reed–Solomon error correction code is used to correct up to 20 bits of error, as the maximum Hamming distance is observed to be 20 for genuine cases. The value of t is set to 20. The parameters for Reed–Solomon code for three datasets DB1, DB2, and DB3 are given in Table 3. It is experimentally observed that the used Reed–Solomon code is able to successfully correct the errors to regenerate any of the stable key strings for all genuine cases. It is also experimentally observed that the error correction capability up to 20 bits cannot regenerate the key strings for impostor cases. It showcases the feasibility of the proposed scheme.

Table 4 Password list

Sequence No	Password
1	J1bˆ7sL@8H
2	efj*3=5QDE
3	XNp+*R72bw
4	aC*55@PkRt
5	#C9RI8_nga
6	@l+9Wj1oYW
7	_jmkZC4:S1
8	$mWH!Fn2e6
9	:h9W.QMfn7
10	-vA8C1Vdr=

Finally, to generate the key having a consistent length of 256 bits for all three datasets (DB1, DB2, DB3), the generated stable bit string s is hashed with the SHA-256 algorithm. This hash value f is considered as the key string from fingerprint biometrics.

Key string generation from password

Moreover, to generate key string p from password for the experiments, ten passwords are randomly generated following the guidelines on password composition (Sect. 3.2.1). Here, the length of each password is considered as 10. These passwords are listed in Table 4.

The generated key string f from the fingerprint biometrics and the password-based key string p are XORed to generate a stable key string k, which is further used as a seed value for key pair generation. It has been observed that the same key string k is generated while combining a specific password with any instance of the 8 fingerprints of a subject. It shows the generation of stable seed value. Thus, in each of the three datasets, all combinations of 10 generated passwords and 80 fingerprints are used to generate the stable seed values for the experiment.

The generated seed value k is used as input to generate two distinct large prime numbers of equal length. Finally, these prime numbers are used to generate the RSA key pair with a length of 2048-bits using the RSA algorithm in 3.2.4.

Experimental setup and computational time

The experiment is run on a Windows 10 computer with an IntelCorei7 processor running at 2.4 GHz, Matlab 2015a, and the Java 1.8 SDK. For extracting minutia points from fingerprint biometrics, commercial software Fingerprint Minutiae Viewer (FpMV) was employed.

The major objective of our approach is the generation of stable key pairs from fingerprint biometrics and password. This process comprises of seven major steps: minutiae extraction, consistent minutiae selection, gray code conversion, stable key string generation, hashed password generation, seed value generation, and key pair generation. It is worth mentioning that the key pair generation requires more time to perform primality tests and integer factorization, which provides more security for the proposed method. In total, it takes around 2.164 s to generate a stable key pair from the user's fingerprint image. However, this time is subjective to the experimental setup we used. Table 5 shows the average execution time for the above-mentioned steps when tested with the benchmark datasets (Table 1).

Security against attacks

This section addresses the protection of fingerprint biometrics, password and private key in the context of proposed approach. Understanding of several attacks reveals that the proposed approach prevents the compromise of fingerprint biometrics, password and the private key. Arguments in favor of the above claim are presented in the following subsections.

No storage of private key

In the proposed method, the generated public key alone is shared with the client to encrypt and send sensitive information. Using the private key the user only can decrypt the sensitive information. Every time, the same cryptographic key pair is obtained using the biometrics and password on the fly. Therefore, the private key need not be stored anywhere, since it can be regenerated on a need basis.

Security of fingerprint template

Only the parity symbols T being generated during the Reed–Solomon encoding process are stored in the database. The parity symbols T do not have any information related to biometrics, and hence the security of the biometric template is ensured against a back-end system compromise, and also it resists record via multiplicity attack, blended substitution attack, and cross-matching attack.

Exhaustive search attack

In this attack, an attacker does not have any information or knowledge about the key generation method, or the ancillary information which is stored in the database. To find the authentic key, the attacker has to use an extensive search. The number of searches depends upon the length of the generated key. If the length of the generated key is L, then the number of searches by the attacker be 2^{L-1}.

Table 5 Computational time of proposed approach

	Steps	Time (sec.)
Key pair generation	Minutiae extraction	0.006
	Consistent minutiae selection	0.3
	Gray code conversion	0.008
	Stable key string generation	0.6
	Hashed password generation	0.02
	Seed value generation	0.03
	Key pair generation	1.2
Total		2.164

Resistance to smart card stolen attack

In [11,47], during key generation phase, the helper data derived from the biometric information is stored in the smart card. During the key retrieval or key regeneration phase, this helper data are used to regenerate or retrieve the same key. If the smart card is stolen or lost, the helper data being stored in the smart card may leak some information related to biometrics. The attacker can also modify the stored helper data. This leads to either the access by an unauthorized person or the denial of a legitimate user. In this work, neither the biometric information, password, nor the generated key is stored. Hence, it resists a smart card stolen attack.

Database compromise

In this attack, an attacker compromises the database. Even if the attacker can access the database, decryption of the data is not possible, since private key is not stored anywhere. The stored parity symbols T do not provide any information related to either the generated cryptographic key pair or the biometric information and password.

Conclusion

In asymmetric cryptography, securing/storing the private key is an arduous task. The proposed two-factor-based stable key pair (public key and private key) generation from fingerprint biometrics and password overcomes this security concern. Usage of two factors to generate the cryptographic key pair differentiates this work from any previous work. It eradicates the need to store a private key. Preferably, the same key pair can be regenerated from fingerprint biometrics and password on a need basis. In this proposed approach, a stable key string is generated from the fingerprint image. The hash value of the stable key string generated from the fingerprint image and the key string generated from hashed password are XORed to generate a stable seed vale. The generated seed value is used to generate the stable key pair.

Another significant feature of this work is the usage of gray code. It has been experimentally shown that the gray code representation of generated key strings significantly reduces the number of mismatching bits between the generated bit strings from the two instances of the same fingerprint compared to a binary code representation. Hence, the Reed–Solomon error correction code successfully corrects the errors resulting from variations in captured images of the same fingerprint. It has also been experimentally concluded that an impostor cannot generate the same key as a genuine user. The proposed work resists Man-in-the-middle attacks, blended substitution, and cross-matching attacks. In the future, other biometric traits such as face, gait, and finger vein could be integrated into this proposed approach.

Funding Not applicable.

Availability of data and materials All data generated or analyzed in this study are included in this published article.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Cimpanu C (2020) https://www.zdnet.com/article/ malicious-chrome-extension-caught-stealing-ledger-wallet-recovery-seeds. Accessed 28 June 2021
2. Gatlan S (2019) https://www.bleepingcomputer.com/news/security/over-435k-security-certs-can-be-compromised-with-less-than-3-000. Accessed 28 June 2021
3. Hao F, Anderson R, Daugman J (2006) Combining crypto with biometrics effectively. IEEE Trans Comput 55(9):1081–1088. https://doi.org/10.1109/TC.2006.138
4. Popa D, Simion E (2017) Enhancing security by combining biometrics and cryptography. In: Proceedings of 9th International Conference on Electronics, Computers and Artificial Intelligence. IEEE, pp 1–7
5. Uludag U, Pankanti S, Prabhakar S, Jain AK (2004) Biometric cryptosystem: issues and challenges. Proc IEEE 92(6):948–960. https://doi.org/10.1109/JPROC.2004.827372
6. Maltoni D, Maio D, Jain AK, Prabhakar S (2003) Handbook of fingerprint recognition. Springer, New York
7. Juels A, Sudan M (2006) A fuzzy vault scheme. Des Codes Crypt 38(2):237–257. https://doi.org/10.1007/s10623-005-6343-z
8. Mahendran RK, Velusamy P (2020) A secure fuzzy extractor based biometric key authentication scheme for body sensor network in Internet of Medical Things. Comput Commun 153:545–552. https://doi.org/10.1016/j.comcom.2020.01.077
9. Juels A, Wattenberg M (1999) A fuzzy commitment scheme. In: Proceedings of ACM 6th conference on computer and communications security, ACM, pp 28–36
10. Adamovic S et al (2017) Fuzzy commitment scheme for generation of cryptographic keys based on iris biometrics. IET Biometric 6(2):89–96. https://doi.org/10.1049/iet-bmt.2016.0061
11. Kausar F (2021) Iris based cancelable biometric cryptosystem for secure healthcare smart card. Egypt Inf J. pp 1–7. https://doi.org/10.1016/j.eij.2021.01.004
12. Panchal G, Samanta D (2018) A novel approach to fingerprint-biometric based cryptographic key generation and its applications to storage security. Comput Electr Eng 69:461–478. https://doi.org/10.1016/j.compeleceng.2018.01.028
13. Suresh K, Pal R, Balasundaram SR (2019) Fingerprint based cryptographic key generation. In: Proceedings of international conference on intelligent data communication technologies and internet of things, Springer, New York, pp 704–713
14. Panchal G, Samanta D (2016) Comparative features and same cryptographic key generation using biometric fingerprints. In: Proceedings of 2nd international conference on advances in electrical, electronics, information, communication and bio-informatics, IEEE, pp 1–5
15. Barman S, Chattopadhyay S, Samanta D (2014) Fingerprint based symmetric cryptography. In: Proceedings of international conference on high performance computing and applications, IEEE, pp 1–6
16. Barman S, Samanta D, Chattopadhyay S (2015) Revocable key generation from irrevocable biometric data for symmetric cryptography. In: Proceedings of 3rd international conference on computer, communication, control and information technology, IEEE, pp 1–4
17. Wang P et al (2021) Biometric key generation based on generated intervals and two-layer error correcting technique. Pattern Recogn 111:1–37. https://doi.org/10.1016/j.patcog.2020.107733
18. Wu Z et al (2018) Generating stable biometric keys for flexible cloud computing authentication using finger vein. Inf Sci 433–434:431–447. https://doi.org/10.1016/j.ins.2016.12.048
19. Rathgeb C, Uhl A (2009) Context-based texture analysis for secure revocable iris biometric key generation. In: Proceedings of 3rd international conference on imaging for crime detection and prevention, IEEE, pp 1–6
20. Kim A, Wang C, Seo S (2020) PCA-CIA ensemble-based feature extraction for bio-key generation. KSII Trans Internet Inf Syst 14(7):2919–2937. https://doi.org/10.3837/tiis.2020.07.011
21. Anees A, Chen Y-PP (2018) Discriminative binary feature learning and quantization in biometric key generation. Pattern Recogn 77:289–305. https://doi.org/10.1016/j.patcog.2017.11.018
22. Monrose F, Reiter M K, Li Q, Wetzel S (2001) Cryptographic key generation from voice. In: Proceedings of IEEE symposium on security and privacy, pp 202–213
23. Sheng W et al (2015) A biometric key generation method based on semi supervised data clustering. IEEE Trans Syst Man Cybern Syst 45(9):1205–1217. https://doi.org/10.1109/TSMC.2015.2389768
24. Xu W et al (2017) Gait-Key: a gait based shared secret key generation protocol for wearable devices. ACM Trans Sensor Netw 13(1):1–27. https://doi.org/10.1145/3023954
25. Sun Y, Lo B (2019) An artificial neural network framework for gait based biometrics. IEEE J Biomed Health Inf 23(3):987–998. https://doi.org/10.1109/JBHI.2018.2860780
26. Wu Y, Lin Q, Jia H, Hassan M, Hu W (2020) Auto-key: using autoencoder to speed up gait-based key generation in body area networks. Proc ACM Interact Mob Wear Ubiquit Technol 4(1):1–23. https://doi.org/10.1145/3381004
27. Sun F, Zang W, Huang H, Farkhatdinov I, Li Y (2021) Accelerometer-based key generation and distribution method for wearable IoT devices. IEEE Internet Things J 8(3):1636–1650. https://doi.org/10.1109/JIOT.2020.3014646
28. Karimian N, Guo Z, Tehranipoor M, Forte D (2017) Highly reliable key generation from electrocardiogram (ECG). IEEE Trans Biomed Eng 64(6):1400–1411. https://doi.org/10.1109/TBME.2016.2607020
29. Moosavi SR et al (2017) Low-latency approach for secure ECG feature based cryptographic key generation. IEEE Access 6:428–442. https://doi.org/10.1109/ACCESS.2017.2766523
30. Roy ND, Biswas A (2020) Fast and robust retinal biometric key generation using deep neural nets. Multimed Tools Appl 79(9):6823–6843. https://doi.org/10.1007/s11042-019-08507-y
31. Joseph T et al (2021) A multi modal biometric authentication scheme based on feature fusion for improving security in cloud environment. J Ambient Intell Hum Comput 12:6141–6149. https://doi.org/10.1007/s12652-020-02184-8
32. Cherupally SK et al (2020) A smart hardware security engine combining entropy sources of ECG, HRV, and SRAM PUF for authentication and secret key generation. IEEE J Solid-State Circ 55(10):2680–2690. https://doi.org/10.1109/JSSC.2020.3010705
33. Yang H, Wu Z (2019) A Biometric key generation method for fingerprint and finger vein fusion. In: Proceedings of international symposium on cyberspace safety and security, Springer, New York, pp 293–300
34. Sarkar A, Singh B K, Bhaumik U (2017) RSA Key Generation from Cancelable Fingerprint Biometrics. In: Proceedings of International Conference on Computing, Communication, Control and Automation. IEEE, pp.1–6
35. Dwivedi R, Dey S, Sharma MA, Goel A (2020) A fingerprint based crypto-biometric system for secure communication. J Ambient Intell Hum Comput 11:1495–1509. https://doi.org/10.1007/s12652-019-01437-5
36. Rathgeb C, Uhl A (2011) A survey on biometric cryptosystem and cancelable biometrics. EURASIP J Inf Secur 3:1–25. https://doi.org/10.1186/1687-417X-2011-3
37. Tams B, Mihailescu P, Munk A (2015) Security considerations in minutiae-based fuzzy vaults. IEEE Trans Inf Forensics Secur 10(5):985–998. https://doi.org/10.1109/TIFS.2015.2392559
38. Panchal G, Samanta D, Barman S (2019) Biometric-based cryptography for digital content protection without any key storage.

Multimed Tools Appl 78:26979–27000. https://doi.org/10.1007/s11042-017-4528-x
39. Barman S, Samanta D, Chattopadhyay S (2015) Approach to cryptographic key generation from fingerprint biometrics. Int J Biometric 7(3):226–248. https://doi.org/10.1504/ijbm.2015.071946
40. Barman S, Samanta D, Chattopadhyay S (2015) Fingerprint based crypto biometric system for network security. EURASIP J Inf Secur 3:1–17. https://doi.org/10.1186/s13635-015-0020-1
41. Barman S, Chattopadhyay S, Samanta D, Panchal G (2017) A novel secure key-exchange protocol using biometrics of the sender and receiver. Comput Electr Eng 64:65–82. https://doi.org/10.1016/j.compeleceng.2016.11.017
42. Sarkar A, Singh BK (2021) A multi-instance cancelable fingerprint biometric based secure session key agreement protocol employing elliptic curve cryptography and a double hash function. Multimed Tools Appl 80:799–829. https://doi.org/10.1007/s11042-020-09375-7
43. Sarkar A, Singh BK (2019) A cancelable biometric based secure session key agreement protocol employing elliptic curve cryptography. Int J Syst Assur Eng Manag 10:1023–1042. https://doi.org/10.1007/s13198-019-00832-7
44. Sarkar A, Singh BK (2019) A cancelable fingerprint biometric based session key establishment protocol. Multimed Tools Appl 78:21645–21671. https://doi.org/10.1007/s11042-019-7426-6
45. Jain A, Hong L, Bolle R (1997) On-line fingerprint verification. IEEE Trans Pattern Anal Mach Intell 19(4):302–314. https://doi.org/10.1109/34.587996
46. Wayne J S, Kenneth K (2014) NBIS NIST Tool. [online] https://www.nist.gov/services-resources/software/fingerprint-minutiae-viewer-fpmv
47. Wang P et al (2021) Biometric key generation based on generated intervals and two-layer error correcting technique. Pattern Recogn 111:1–39. https://doi.org/10.1016/j.patcog.2020.107733
48. Stephen BW, Vijay KB (eds) (1999) Reed-Solomon codes and their applications. IEEE Press
49. Turan MS, Barker E, Burr W, Chen L (2010) Recommendation for password-based key derivation. NIST Special Publication 800-132. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
50. Yao FF, Yin YL (2005) Design and analysis of password-based key derivation functions. IEEE Trans Inf Theory 51(9):3292–3297. https://doi.org/10.1109/tit.2005.853307
51. Zhou J et al (2012) On the security of key derivation functions in office. In: Proceedings of international workshop on anti-counterfeiting, security, and identification, IEEE, pp 1–5
52. Grassi PA et al (2017) Digital identity guidelines: authentication and lifecycle management. Special Publication (NIST SP)-800-63B. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf
53. O'neill ME (2009) The genuine sieve of Eratosthenes. J Funct Program 19(1):95–106. https://doi.org/10.1017/S0956796808007004
54. Agrawal M, Kayal N, Saxena N (2004) Primes is in P. Annals of Mathematics. 160(2):781–793. https://doi.org/10.4007/annals.2004.160.781
55. Albrecht M R, Massimo J, Paterson KG, Somorovsky J (2018) Prime and Prejudice: Primality Testing Under Adversarial Conditions. In: Proceedings of ACM SIGSAC conference on computer and communications security, pp 281–298
56. FVC 2002 Database. [online] http://bias.csr.unibo.it/fvc2002/databases.asp
57. Kaur M, Sofat S (2016) Secure fingerprint fuzzy vault using hadamard transformation to defy correlation attack. In: Proceedings of 6th International Symposium on Embedded Computing and System Design, IEEE, pp 122–126