World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:12, No:10, 2018

Fingerprint Image Encryption Using a 2D Chaotic Map and Elliptic Curve Cryptography

D. M. S. Bandara, Yunqi Lei, Ye Luo

Abstract**—**Fingerprints are suitable as long-term markers of human identity since they provide detailed and unique individual features which are difficult to alter and durable over life time. In this paper, we propose an algorithm to encrypt and decrypt fingerprint images by using a specially designed Elliptic Curve Cryptography (ECC) procedure based on block ciphers. In addition, to increase the confusing effect of fingerprint encryption, we also utilize a chaotic-behaved method called Arnold Cat Map (ACM) for a 2D scrambling of pixel locations in our method. Experimental results are carried out with various types of efficiency and security analyses. As a result, we demonstrate that the proposed fingerprint encryption/decryption algorithm is advantageous in several different aspects including efficiency, security and flexibility. In particular, using this algorithm, we achieve a margin of about 0.1% in the test of Number of Pixel Changing Rate (NPCR) values comparing to the-state-of-the-art performances.

Keywords**—**Arnold cat map, biometric encryption, block cipher, elliptic curve cryptography, fingerprint encryption, Koblitz's Encoding.

I. Introduction

BIOMETRIC is defined as a unique, measurable, biological characteristic or quality for automatic identification and verification of the identity of a human being. Analyzing of these biological characteristics has become known as the science of biometrics. These days, biometric technologies are usually used to analyze human characteristics for security purposes. The exchange of these biometric data over unsecured network channels makes it at risk to the violation of security by illegal access. To overcome this challenge, in this paper, we propose an efficient and secure encryption scheme which behaves with remarkable complexity and is difficult to predict and hack for intruders.

Combining biometrics and cryptography has the potential to provide higher assurance of system security. A security scheme that utilizes both biometrics and crypto keys is known as biometric encryption. In this paper, we use fingerprint images as biometric data. Moreover, the cryptographic technique that we have used in this paper is specially designed for fingerprint encryption based on block ciphers and the general ECC which is a public key cryptosystem originally developed by Koblitz [1] and Miller [2]. After 2004, ECC becomes widely used for information security as an encryption tool. Many researchers have concluded that the difficulty of solving Elliptic Curve Discrete Logarithmic Problem over finite fields is very hard with respect to a proper key size, and this property enables ECC to become aremarkable tool for encryption and decryption processes under various circumstances. In particular, for this research, we use three different prime fields with respect to three different bit sizes of 128, 256, and 512 bits as the underlying finite fields of ECC. Fingerprints are encrypted on using ECC over these three prime fields. Furthermore, for increasing the overall performance of our method, we also incorporate ACM for permuting pixel locations as an initial step of encrypting the fingerprint images.

Finally, we perform various analyses to our algorithm to test the efficiency and security performance of our algorithm including keyspace analysis, NPCR analysis, Unified Average Charging Intensity (UACI) analysis, speed analysis, histogram analysis, and known plaintext and chosen plaintext attacks analysis. Our method has shown great results during all these analyses.

II. Mathematical Preliminaries

The most important mathematical operations regarding ECC are briefly explained in the following.

An elliptic curve over a prime number p is an algebraic group defined by a cubic equation with the form,

$$y^2 = \{x^3 + ax + b\}\, modp$$

where $4a^3 + 27b^2 \neq 0\ (modp)$.

Point addition, subtraction, multiplication and doubling are the key arithmetic operations on an elliptic curve. Suppose that points P, Q and R are all on the elliptic curve, i.e. their x- and y-coordinates satisfy the above elliptic curve equation.

A. Point Addition

Point R with coordinates $(x_3, y_3)$is a sum of Point $P(x_1, y_1)$ and Point $Q(x_2, y_2)$under the group addition of the elliptic curve. Then, we have the following formulas:

$$P(x_1, y_1) + Q(x_2, y_2) = R(x_3, y_3)$$
$$x_3 = (\lambda^2 - x_1 - x_2)modp$$
$$y_3 = (\lambda(x_1 - x_3) - y_1)\, modp$$

where $\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)} modp$.

B. Point Substraction

For subtraction, we mirror the coordinates of particular point

D. M. S. Bandara is with the Department of Computer Science, School of Info. Sci. & Eng., Xiamen University, China; Dept. of Mathematical Sciences, FAS, Wayamba University of Sri Lanka (corresponding author, phone: +0094713934470/+86-13276028502; e-mail: bandaradms@gmail.com).

Lei, Yunqi is with the Department of Computer Science, School of Info. Sci. & Eng., Xiamen University, China (e-mail: yqlei@xmu.edu.cn).

Luo Ye is with the Department Dept. of Cyberspace Security, School of Info. Sci. & Eng., Xiamen University, China (e-mail: luoye@xmu.edu.cn).

along the x-axis and apply the above addition operation to compute subtraction as follows

$$P(x_1,y_1) - Q(x_2,y_2) = P(x_1,y_1) + Q(x_2,-y_2)$$

C. Point Doubling

To compute point doubling, we identify point P and point Q in the point addition formula and get

$$P(x_1,y_1) + Q(x_2,y_2) = R(x_3,y_3)$$
$$x_3 = (\lambda^2 - 2x_1) mod p$$
$$y_3 = (\lambda(x_1 - x_3) - y_1) mod p$$

where, $\lambda = \frac{(3x_1^2+a)}{2y_1} mod p$.

D. Point Multiplication

Point multiplication of an integer n and a point p is defined as $np=p+p+\ldots+p$ (n times). Note that there exist fast algorithms to compute point multiplication based on point doublings.

III. Literature Review

For the security of fingerprint image information, various techniques have been developed in this emerging research area recently.

According to review article of Jain et al. [3], template protection approaches were classified into three main categories: feature transformation, biometric cryptosystem, and hybrid systems. Panchal and Samanth [4] proposed an encryption method by extracting statistical feature to generate a codeword. Then, by using Reed-Solomon encoding, they convert that codeword to the key for their encryption scheme which means the key is generated from the original fingerprint image. Moujahdi et al. [5] proposed an approach to protect fingerprint template. They exploit the information provided by the extracted minutiae to construct a new representation based on special spiral curves which are stored in the database system to be used for recognition. 2D chaotic sequences from multi-scroll chaotic attractors were used for encryption by Han et al. [6]. Encryption technique using minutiae-based transformation was proposed by Haiyong and Hailiang [7]. In this algorithm, a circular region around each minutia is constructed, and non-invertible transformation is applied to all minutiae regions while all transformations are stored in the database. Zhao and Yan [8] proposed a scheme combining with shuttle operation and nonlinear dynamic chaos system. Mehta et al. [9] proposed a method based on chaotic theory using Arnold and Henon maps for generating cipher image. Hsiao and Lee [10] proposed to combine four chaotic systems for encryption including two 1-D and two 3-D chaotic systems.

IV. Problem Statement

All these algorithms used large key sizes. One of the biggest advantages of using ECC for encryption is that ECC usually requires much smaller key sizes to achieve an analogous security level compared with the above schemes and other existing techniques. On the other hand, we almost have an unlimited source of elliptic curves which can be used for ECC. Actually, ECC Brain Pool [11] and National Institute of Standards and Technology (NIST) provide various recommended and secured elliptic curve parameters of different bit sizes for researchers. Fingerprint images that we have used here are obtained from Biometric Ideal Test (BIT) [12].

Some of the above-mentioned methods use random key generation techniques where keys are usually generated from pixel values. The problem arising here is that random key generation cannot generate the same key in two or more sessions which means their methods are not capable of meeting the requirements in many application situations.

Many standard cipher systems such as IDEA and RSA are not suitable for practical fingerprint encryption because of their relatively large key sizes and high computational complexity.

In this work, we propose a method to overcome the above issues by developing an adapted version of ECC procedure which inherits the advantages of being fast and secure of the general ECC schemes. Moreover, to enhance the overall confusion effect during encryption, we propose to add a pre-processing technique before applying the main ECC encryption algorithm. In particular, we use ACM for initial permutation which shuffles pixels in the original fingerprint images before applying ECC and this enhances remarkably the overall performance of our algorithm. Furthermore, in this research, we use big prime fields on which we can deal with very big numbers at one time. This also enables us to incorporate block cipher techniques to reduce the computation time for encryption and decryption and increase the flexibility of our scheme.

V. Proposed Method

Since fingerprint encryption is a special category in the image encryption field, we should concern about image encryption processes for this special type of images. Our algorithm can be stated in a few main steps:applying ACM, selecting prime fields and coefficients for elliptic curves, dividing the image into blocks, key generation, and the final overall encryption and decryption algorithms.

A. Arnold Cat Map (ACM)

ACM is a 2D invertible linear transformation of the real torus $\mathbb{R}^2/\mathbb{Z}^2$ to itself with a chaotic behavior. The discrete analogue, also called Discrete Cat Map (DCM), can be used to do permutations of pixels in a digital image. In particular, DCM has the Poincare Recurrence behavior, i.e. after a certain number of iterations (the number depending on the image size) of ACM, the original image will be restored.

More precisely, ACM is defined as the transformationΓ : $\mathbb{R}^2/\mathbb{Z}^2 \rightarrow \mathbb{R}^2/\mathbb{Z}^2$ where, in matrix notation,

$$\Gamma\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} mod\, 1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} mod\, 1$$

Note that $det \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} = 1$, and thus, ACM is area-preserving.

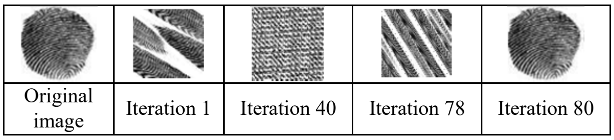
Now let us consider an image with $N \times N$ pixels and apply DCM to this image. Let x and y be the coordinates of the pixels.

Then, the image recurrence relation can be expressed as follows with respect to $modN$.

$$n = 0: \quad \Gamma^0(x,y) = Input\ Image\ (x,y)$$
$$n = 1: \quad \Gamma^1\ (x,y) = \Gamma^0((2x+y)\ mod\ N, (x+y)\ mod\ N\ )$$
$$\vdots$$
$$n = k: \quad \Gamma^k\ (x,y) = \Gamma^{k-1}\ ((2x+y)\ mod\ N, (x+y)\ mod\ N\ )$$
$$\vdots$$
$$n = mOutputImage\ (x,y) = \ \Gamma^m\ \ (x,y)$$

In particular, it can be shown that after N iterations, the image will be restored to the original one (Poincare recurrence).

Fig. 1 shows the results of applying ACM to a fingerprint image (80×80 pixels) with different iteration times. For each iteration, the image is stretched and sliced into pieces which are then recombined into a new image. Note that, in this example, after applying ACM 80 times, the image is restored to the original one.



Original image | Iteration 1 | Iteration 40 | Iteration 78 | Iteration 80

Fig. 1 A few iterations of ACM

In our image encryption, we apply ACM for a fixed number of times. For example, to get a good confusion and diffusion in pixel scrambling, we may choose to apply ACM foraround 40iterations for an 80×80 image. Moreover, since the number of iteration is fixed, we simplify our computation by using the following formula (called simplified ACM computation):

$$n = 0: \quad \Gamma^0(x,y) = InputImage(x,y)$$
$$\vdots$$
$$n = m: \quad \Gamma^m\ (\begin{bmatrix} x \\ y \end{bmatrix}) = \Gamma^0(\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}^m \begin{bmatrix} x \\ y \end{bmatrix} mod\ N)$$

Actually, using the simplified ACM can tremendously reduce the overall encryption/decryption computation time (see the speed analysis part of Section VI).

B.Prime Fields and Curve Coefficients

For bit sizes of 128-bit, 256-bit and 512-bit, prime fields of those bit sizes and other elliptic curve parameters are respectively chosen from some standard elliptic curves given by ECC Brain pool [11].

The domain parameters for a selected elliptic curve $y^2 = \{x^3 + ax + b\}\ mod\ p$is ($p$,$a$,$b$,$G$,$n$) where p is the prime number for the underlying prime field of the elliptic curve, a and b are the curve coefficients, G is called generator which is a point of the curve, n is the order of generator in the group of the elliptic curve. For example, the elliptic curve parameters that we choose for 128-bit class are as follows:

(p=2552552552232552552552552552552552552552255255259,
a=255255255253255255255255255255255255255255255255252
,b=232117121193161212446121636153604423894211,
G=(223124782139137155451240961241654491134,
122451721311491862541776022184145222215168),
n=255255255254000011716313271445616121)

C. Dividing the Image into Blocks

We also employ block cipher technique in our encryption. In particular, we only need to use the simplest Electronic Code mode of block cipher, since in the step of applying ACM, the pixels have already been scrambled.

The block size depends on the selected prime number in Step B. For each pixel, we choose the pixel value from 0-255 which is of 8 bits. The following table gives an example of the block sizes that we choose with respect to corresponding bit sizes. Note that other block sizes (e.g., 15 pixels of 3×5 for the 128bit case or 63 pixels of 1×63 for the 512bit case) are also optional as long as they are compatible with the bit size. For each block, we combine and concatenate binary pixel values in this block into one big integer, while it should be considered as an element of the prime field for the chosen elliptic curve of the corresponding bit size.

TABLE I
AN EXAMPLE OF BLOCK SIZES

	Block size	# pixels
128bit	4×4	16
256bit	4×8	32
512bit	8×8	64

D.Elliptic Curve Key Generation

As in general public key cryptosystems, we need to generate a private key for decryption and a public key for encryption. For a selected elliptic curve with parameter (p,a,b,G,n) (Step B), a private key is an integer $K_{pvt} = d$ randomly chosen from the interval $[1, n-1]$, while the corresponding public key is $P_B = d\ G$. We summarize this key generation in Algorithm 0.

```
Algorithm 0: Key generation
Input: Elliptic curve parameters $(p, a, b, G, n)$
Output: Public key $P_B$ and private key $K_{pvt}$
1. Select randomly $d \in [1, n-1]$.
2. Compute point multiplication $dG$of $d$ and $G$.
3. Let $K_{pvt} = d$ and $P_B = d\ G$.
```

E. Encryption and Decryption

The overall encryption algorithm is summarized in Algorithm 1. Our input includes the selected elliptic curve (Step B), a public key generated in Algorithm 0 and the fingerprint image for encryption. We first apply ACM (see Step A for details) for an initial chaotic permutation of the image pixels. Then, we divide the image matrix into blocks and for each block a big integer is generated. Next, we want to apply ECC to each big integer using the same public keyP_B. In particular, we combine Koblitz's Encoding Method [13] which is used to map each big integer to a point on the elliptic curve and a formal approach of ECC encryption for general image encryption [14]. The output is the cipher data composed of a bunch of cipher pointsY_i on the elliptic curve, one for each block. In addition, we also include an extra point X on the elliptic curve in the cipher data. X is independent of the input image but will be used in the decryption procedure. Moreover, for the purpose of illustration, we can generate the cipher image from the cipher data.

```
Algorithm 1: Fingerprint image encryption
Input: Elliptic curve with parameters $(p, a, b, G, n)$, public key $P_B$and
a fingerprint image of size $N \times N$.
Output :(a) Cipher data$(m, K', X, \{Y_i\}_{i=1,\ldots,L})$wherem is a number in
$[0, N-1]$, K' is a number in $[1, n-1]$, $X$ and $Y_i$ are points on the
elliptic curve, and $L$ is the number of blocks; (b) A cipher image of size
$N \times N$ generated from $\{Y_i\}_{i=1,\ldots,L}$.
1.  Image → Pixel matrix.
2.  Apply ACM to the pixel matrix for $m$ iteration times (details in
    Step A).
3.  Dividing the pixel matrix derived in Step 2 into
    $L$blocks$B_1, \ldots, B_L$(Step C).
  For each block $B_i$, do Step 4-8:
4.  Generate a big integer$M_i$for each block $B_i$ by concatenating the
    binary pixel values of all the pixels in $B_i$.
5.  $Point$ 1 = A point in the elliptic curved obtained by using
    Koblitz's Encoding Method [13] to embed the big integer$M_i$in its
    x-coordinates:
a.  Choose an auxiliary parameter$K' \in [1, n-1]$. (e.g. we may
    choose $K' = 40$ which is fixed throughout the
    encryption/decryption process.)
b.  Let$x = M_i K'$. If $x^3 + ax + b$ is a square mod $p$, then we let $y$ be
    a square root and (x,y) be the coordinates of $Point$ 1.
c.  If $x = M_i K'$ does not make $x^3 + ax + b$ a square mod $p$, then let
    $x = M_i K' + j$ for j going from 1 through $K' - 1$ until $x^3 + ax +$
    $b$ first becomes a square mod $p$. Then let $y$ be a square root of
    $x^3 + ax + b$mod $p$ and let (x,y) be the coordinates of $Point$ 1.
6.  Select random integer $K \in [1, n-1]$ which is fixed throughout
    the encryption process.
7.  $Point$ 2 = point multiplication of $K$ and $P_B$.
8.  Compute $Y_i$ = point addition of $Point$ 1 and $Point$ 2 .
9.  Compute $X$ = point multiplication of $K$ and$G$.
10. Assemble $X$ and all $Y_i$'s into the cipher data: $(X, \{Y_i\}_{i=1,\ldots,L})$
11. Cipher image can be generated by conversely mapping the
    x-coordinate of the cipher point $Y_i$ to pixel values for all blocks $B_i$
```

The decryption algorithm is summarized in Algorithm 2. For input, we will use the same elliptic curve as in encryption, the private key generated in Algorithm 0, and the cipher data. In this algorithm, Step 1 and 2 come from formal ECC decryption.

Note that Y'_i obtained in Step 2 of Algorithm 2 is identical to $Point$ 1in Step 5 of Algorithm 1. Then, by Koblitz' Decoding, we convert Y'_i back into a big integer M_i which is then subdivided into pixel values to generate the block B_i. By assembling all the block, we get an $N \times N$image. Finally, by applying ACM of $N - m$ iterations, we obtain the decrypted $N \times N$ image.

```
Algorithm 2: Fingerprint image decryption
Input: Elliptic curve parameter $(p, a, b, G, n)$, private key $K_{pvt}$ and
cipher data$(m, K', X, \{Y_i\}_{i=1,\ldots,L})$.
Output: Original fingerprint of size $N \times N$.
1.  $Point$ 3 = point multiplication of $K_{pvt}$ and X.
  For each $i = 1, \ldots, L$, do Step 2 and 4:
2.  $Y'_i$ = point subtraction of $Y_i$ and $Point$ 3.
3.  Koblitz's Decoding: convert$Y'_i$back into a big integer $M_i$ by
    letting $\mathrm{M_i}$ be the greatest integer less than or equal to $\mathrm{x_i}/\mathrm{K}'$ where
    $\mathrm{x_i}$ is the x-coordinate of $Y'_i$.
4.  Write $Y'_i$ as a binary integer, subdivided into pixel values and
    combine orderly the pixels to generate the corresponding block
    $B_i$.
5.  Assemble all the blocks $B_i$'s into an $N \times N$ matrix of pixels.
6.  Apply ACM on generated image for $N - m$ iteration times the
    decrypted fingerprint image is obtained.
```

VI. Discussion and Security Analysis of the Results

The implementation is performed on Core i5 CPU 3.20GHz desktop with 8GB RAM using Mathematica (version 11).

As an example of the results, some fingerprint images, cipher images after encryption and their corresponding histograms are shown in Fig. 2. Meanwhile, we also apply the encryption to the benchmark Lena's image as shown in Fig. 2. Note that in these results, the input images have different types of histograms. In particular, the first fingerprint image has its histogram dominated by the black and white part which is very common to general fingerprint images; the second fingerprint image still has a dominant white part, while the black ridges become much lighter in grayscale which is also commonly happened case in fingerprint scanning; the Lena image has a typical histogram of a general image which has curved distribution dominated by the middle part. However, no matter what type of histogram the input image has, the corresponding cipher image is always white-noise-like with a flat distribution in its histogram. We will get to more detailed discussions in the following security analysis of our encryption/decryption algorithm.

Security analysis of a cryptographic algorithm is crucial to ensure the strength of the used technique. To ensure the security of our fingerprint encryption method, several widely used factors are considered here, including keyspace analysis, NPCR, UACI, speed analysis, histogram analysis, and known plaintext and chosen plaintext attacks analysis. In particular, we discuss the results from our full algorithm together with a comparison to results from a partial algorithm where ACM is not performed.

Keyspace Analysis

To find the key used to encrypt image the fundamental way of cryptanalysis is brute-force attack. As stated in Algorithm 2, the private key use for decryption is chosen from [*1*,*n-1*] where n is the order of the generator G which is typically about the size of the underlying prime field. This means that for the used bit sizes of 128-bit, 256-bit and 512-bit, the keyspace is of 128-bit, 256-bit and 512-bit respectively while 128-bit keyspace is generally considered as secure with respect to brute-force attack.

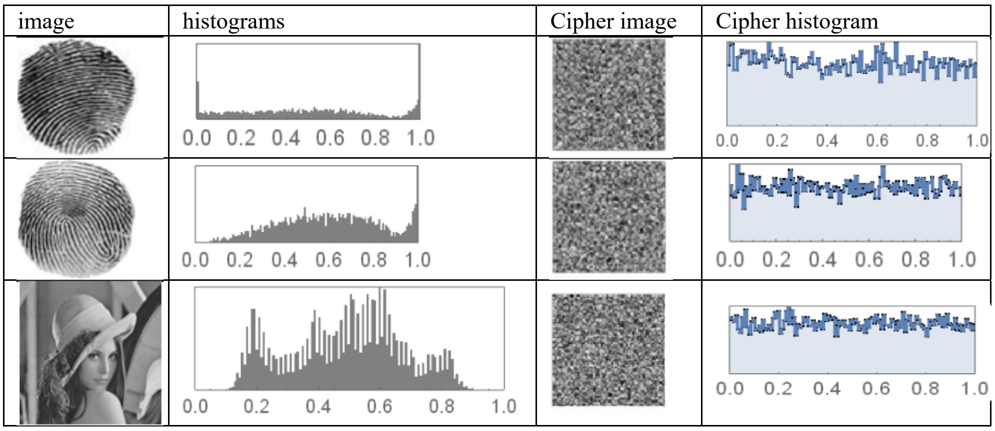


Fig. 2 Some encryption results

NPCR and UACI Tests

An important requirement of the security of an encryption technique is that small changes of the plain image should result in a highly changed cipher image. For example, this requirement was demonstrated and measured by Wu et al. [15] using the standard tool, i.e. NPCR and UACI defined as

$$NPCR{:}\, N(C_1, C_2) = \sum_{i,j} \frac{D(i,j)}{T} \times 100\%$$
$$UACI{:}\, U(C_1, C_2) = \sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{F.T} \times 100\%$$

where $C_1(i,j) and C_2(i,j)$ are pixel values at (i,j) location before and after encryption, T - total number of pixel, F - maximum encrypted value (for gray level image is 255) and $D(i,j)$ marks whether the pixel at (i,j) changes, i.e.,

$$D(i,j) = \begin{cases} 0, & if\, C_1(i,j) = C_2(i,j) \\ 1 & if\, C_1(i,j) \neq C_2(i,j) \end{cases}.$$

We do such an analysis on our algorithm by averaging values over 53 randomly chosen fingerprint images with results shown in Table II.

TABLE II
NPCR AND UACI TESTS ON OUR SCHEME

Bit Size	ACM	NPCR (mean)(%)	UACI (mean)(%)
128	No	99.65	33.60
	Yes	99.73	33.41
256	No	99.62	33.12
	Yes	99.71	33.64
512	No	99.66	33.07
	Yes	99.71	33.20

For all bit sizes, our algorithm consistently gives NPCR values of around 99.65% without ACM and values of more than 99.7% with ACM. This means incorporating ACM in encryption enhances the security performance of the algorithm. In addition, we compare our results with a few reported works shown in Table III where the typical NPCR that they can achieve is around 99.6%. While the UACI values of our work and others' work are about the same level, our algorithm has a better performance than the-state-of-art with a margin of NPCR value for ~0.05% without ACM and ~0.1% with ACM.

TABLE III
NPCR AND UACI VALUES OF A FEW REPORTED SCHEMES

Method	NPCR (mean)%	UACI (mean)%
[16] (Wu et al. 2012)	99.60	33.51
[10] (Hsiao et al. 2015)	99.61	33.46
[17] (Toughi et al. 2017)	99.60	33.48
[18] (Wang et al. 2018)	99.59	33.45

Speed Analysis

The computational time of encryption and decryption depends on various factors such as the size of image, the operating systems, performance of the computer, programming languages, and the algorithms themselves etc. We have implemented our algorithm in Mathematica version 11, execution time of encryption and decryption for fingerprint images is given by the following table whose values are taken from the average of the encryption /decryption of 53 fingerprints. Note that as the bit size increases (and correspondingly the keyspace increases which is generally considered to be more secure), the encryption time only increases marginally.

TABLE IV
COMPUTATIONAL TIME

Bit Sizes	Encryption Time (sec.)	Decryption Time (sec.)
128	0.5716	0.10937
256	0.6390	0.15625
512	0.6800	0.25

In Fig. 3, we show the computation time of the 512-bit encryption for all the 53 images. The blue curve, green curve, and yellow curve show the computation time for encryption without ACM, with simplified ACM (see Section V.*A*) and with iterated ACM respectively. Note that by simply replacing the iterated ACM by simplified ACM, the computation time is tremendously reduced. Moreover, for all cases, it can be observed from the figure that the proposed algorithm gives very small variations of computation time which means our algorithm is a stable scheme. In particular, the time variation of the green curve (with-simplified-ACM case) is 0.09375 sec.s. For the 128-bit and 256-bit cases, the time variations are 0.0625 sec.s and 0.03125 sec.s respectively which are even smaller.

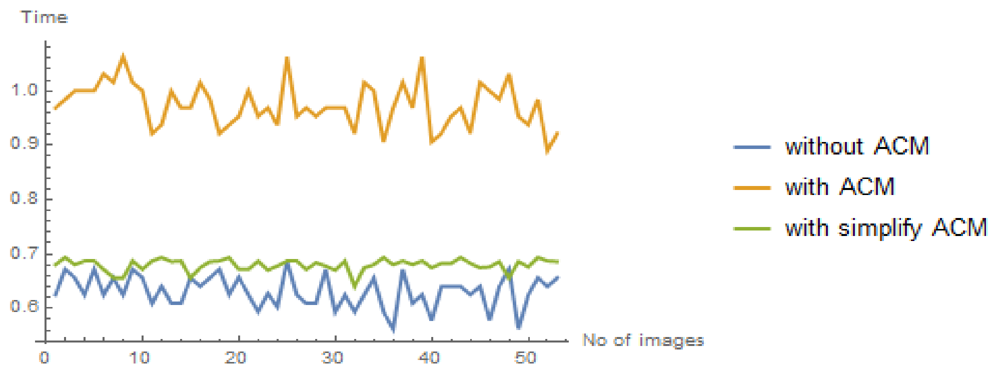


Fig. 3 Time variation of encryption

Histogram Analysis

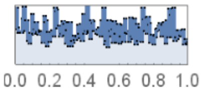
Good cipher images are generally required to have a close-to-uniform distribution of pixel values which is white-noise-like. As having been shown in Fig. 2, we can see that even though the original images have different types of image histograms, the cipher image histograms are always close to a uniform frequency distribution. No information is directly readable from the cipher images since they always appear as white noises.

To ensure that the cipher image does not encode certain location information, we can make subdivision of cipher image into blocks and do histogram analysis on each one of the blocks. In the following example shown in Fig. 4, we do not just derive the overall histogram of the cipher image of a plain fingerprint, but we divide the cipher image into 3 × 3 blocks and derive the histograms of all cipher blocks. It can be observed that all of them have a close to a uniform frequency distribution.

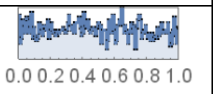
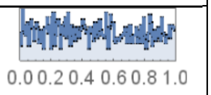
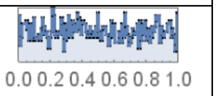
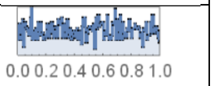
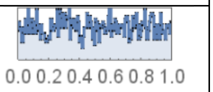
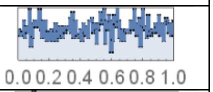
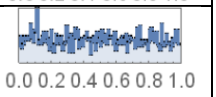
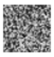
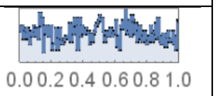
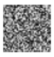
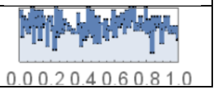
Known Plaintext and Chosen Plaintext Attack Analysis

Known plaintext and chosen plaintext attacks have been used to crack some encryption algorithms. Here, we show some results from preliminary known plaintext and chosen plaintext attacks. For example, some intruders have used all black and all white images for the searching pattern of the algorithm. In Fig. 5, we show the cipher images of pure white and pure black plaintext images which are both white-noise-like with no informative patterns visible. Therefore, we expect our algorithm to be highly resistant to these types of attacks.

We obtained similar results during encryption of positioning images of pure dark and pure white image as shown in Fig. 6.

Plain image	Cipher image	Cipher image histogram
		

(a)

Cipher block Index	Cipher block	Histogram
(1,1)		
(1,2)		
(1,3)		
(2,1)		
(2,2)		
(2,3)		
(3,1)		
(3,2)		
(3,3)		

(b)

Fig. 4 (a) A fingerprint image, its cipher image and the histogram of the cipher image. (b) The nine sub cipher blocks of the cipher image and their corresponding histograms

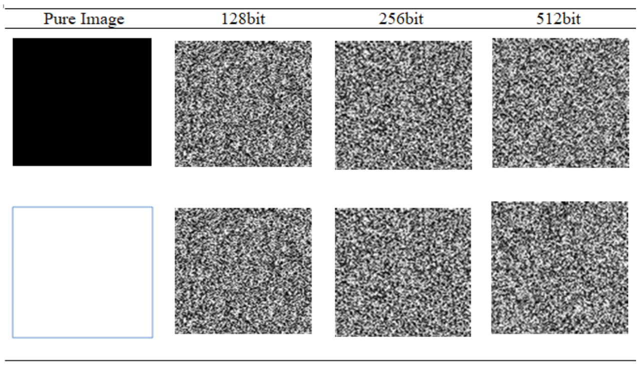


Fig. 5 Pure dark image and pure white encryption

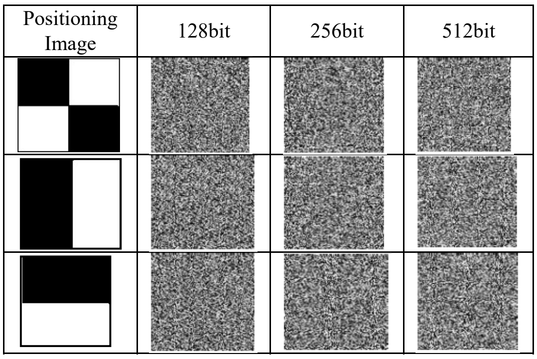


Fig. 6 Positioning image encryption

Furthermore, here we show some results from pure image attack. Suppose that the input plaintext image is a pure image of grayscale X in [0,255] while the corresponding normalized grayscale is $X/256$. We compute the UACI value $Y\%$ of the cipher image based on our encryption algorithm. Samples of computing $(\frac{X}{256}, Y\%)$ are shown as blue dots in Fig. 7. It is clearly shown that the sample points are in general very close to the red curve computed from a theoretical "ideal" encryption. Here an ideal encryption means that, for any pixel in the cipher image, the pixel value of that pixel the cipher image has a uniform distribution which is independent from the pixel value at the same position of the input plaintext image. Therefore, the red curve can be computed theoretically as follows:

$$Y\% = \int_0^1 \int_0^1 P(u,v)|u-v|dudv = \int_0^1 \int_0^1 P_u(u)P_v(v)|u - v|dudv = \int_0^1 \int_0^1 \delta_x(u)|u-v|dudv = \frac{x^2+(1-x)^2}{2}$$

where $x = X/256$ is the normalized grayscale of the input image. Note that in the above computation, $P_u(u)$ is the histogram distribution of the input image which is the delta function at the normalized gray scale x of the input image, $P_v(v)$ is the histogram distribution of the "ideal" cipher image which is uniformly 1 for all $v \in [0,1]$, and $P(u,v) = P_u(u)P_v(v)$ since the grayscale distribution of the "ideal" cipher image is independent from the grayscale distribution of the input image by assumption. As shown in Fig. 7, the results (blue dots) from our encryption algorithm provide a remarkable fit to the red curve derived from a theoretically "ideal" encryption method.

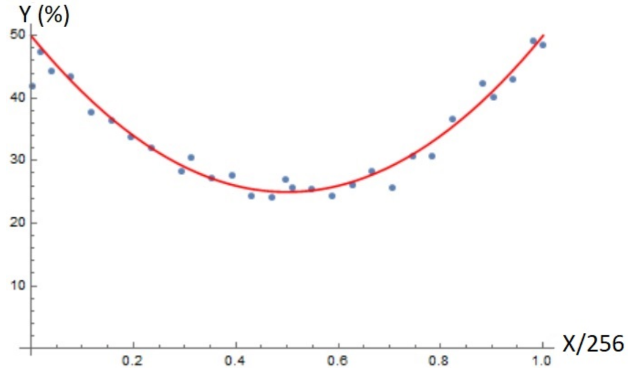


Fig. 7 The UACI value ($Y\%$) vs the normalized grayscale ($X/256$) of the pure image. The red curve is the curve derived from ideal encryption and the blue dots represent the samples using our encryption algorithm

VII. Conclusions

In this paper, a method is proposed for fingerprint encryption. Our algorithm is performed by combining an ECC type of image encryption and ACM. We have also done several conventional tests of security analysis on the proposed encryption/decryption scheme. In particular, the ACM procedure provides a 2D shuffle of the image pixels with a chaotic behavior which enhances the confusion effect of encryption. This is indicated by a margin of about 0.1% in the test of NPCR values comparing to existent reported work. The specially designed ECC procedure which incorporates block cipher techniques in our algorithm utilizes the advantages of general ECC techniques, e.g. relatively smaller key sizes, high speed process, and high security. In addition, our algorithm provides options in choosing key sizes and block sizes which enable a flexible balance of computation resources and meeting higher security requirements. We have applied various thorough security analyses to our algorithm with great results. We conclude that the proposed algorithm has a great performance in both efficiency and security which is highly practical for fingerprint data storage and transmission through unsecure networks.

References

[1] N. Koblitz, "Elliptic Curve Cryptography" Mathematics of Computation, AMS, vol. 48, no. 177, pp. 203-208, 1987.

[2] V. Miller, "Uses of Elliptic Curve in Cryptography", Advanced in Cryptology, Springer-Verlag vol. 85, pp. 417-426, 1986.

[3] A. K. Jain, K. Nandakumar, A. Nagar, "Biometric template security", Advances in Signal Process, EURASIP journal, vol. 2008, pp. 1-17, 2008.

[4] G. Panchal and D. Samanta, "A novel Approach to Fingerprint Biometric-Based Cryptographic Key Generation and its Application to storage security", Computer and Electrical Engineering 000, Elsevier, pp. 1-18, 2018.

[5] C. Moujahdi, G. Bebis, S. Ghouzali and M. Rizza, "Fingerprint shell: Secure representation of fingerprint template", Pattern Recognition Letters 24, Elsevier, pp. 189-196, 2014.

[6] F. Han, J. Hu, X. Yu and Y. Wang, "Fingerprint image encryption via multi-scroll chaotic attractors", Applied Mathematics and Computation 185, Elsevier, pp. 931-939, 2007.

[7] Haiyong Chen and Hailiang Chen, "A novel algorithm of fingerprint

encryption using minutiae-based transformation", Pattern Recognition Letters 32, Elsevier, pp. 305-309, 2011.

[8] S. Zhao, H. Li and X. Yan, "A secure and efficient fingerprint image encryption scheme", The 9th International Conference for Young Computer Scientists, IEEE Computer Society, P.R.C, pp. 2803-2808, 2008.

[9] G. Mehta, M. K. Dutta, J. Karasek and P. S. Kim, "An efficient and lossless fingerprint encryption algorithm using Henon Map & Arnold Transformation", International Conference on Control Communication and Computing, IEEE, pp. 485-48, 2013.

[10] H-I Hsiao and J. Lee, "Fingerprint image cryptography based on multiple chaotic system", Signal Processing 131, Elsevier, pp. 169-181, 2015.

[11] ECC Brainpool Standard Curves and Curve Generation v.1.0, 19.10.2005

[12] Institute of Automation, Chinese Academy of Sciences(CASIA), Biometric ideal test, Accessed 10 January 2017, <http://biometrics.idealtest.org>.

[13] O. Reyad and Z. Kotulski, Image "encryption using Koblitz's encoding and new mapping method based on elliptic curve random number generator", Springer International Publishing, Switzerland, 2015, pp. 34-45.

[14] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer, 2004.

[15] Y. Wu, P. Noonan and S. Agaian, "UPCR and UACI randomness test for image encryption", Cyber Journal: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunicstion (JSAT), pp. 31-38, April Edition, 2011.

[16] Y. Wu, G. Yang, H. Jin and J.P. Noonan, "Image encryption using the two dimensional logistic chaotic map", Journal of Electronic Imaging, vol. 21(1), 013014, March, 2012.

[17] S. Toughi, M. H. fathi and Y. A. Sekhavet,, "An image encryption scheme based on elliptic curve pseudo random and Advanced encryption system", Signal processing, vol. 141, Elsevier, pp. 217-227, 2017.

[18] X. Wang, X. Zhu and Y. Zhang, "An image encryption algorithm based on Josephus Traversing and mixed chaotic map", IEEE, 2018.