Information Sciences Letters
An International Journal

Secure Hill Cipher Modification Based on Generalized Permutation Matrix SHC-GPM

A. Y. Mahmoud[1] **and Alexander G. Chefranov**[2]
[1]Department of Information Technology, Faculty of Engineering and Information Technology, Al-Azhar University- Gaza
[2]Computer Engineering Department, Eastern Mediterranean University, North Cyprus, Turkey
Email: ahmed@alazhar.edu.ps, Alexander.chefranov@emu.edu.tr

Received: 1 Jan. 2012, Revised: 12 May 2012, Accepted: 16 Jun. 2012

Abstract: Secure Hill cipher (SHC) modification based on dynamically changing generalized permutation matrix, SHC-GPM is proposed. It provides better security than that of SHC due to the significantly larger number of non-repeatedly generated keys (2^{48} times greater for the parameters used in the conducted experiments). SHC-GPM is shown to be robust against the brute-force, statistical attacks, and is resistant also to known plaintext ciphertext attack (KPCA) due to dynamic encryption key matrix generating. The proposed modification is applied for colour images encryption. Experimental results show that the proposed modification is significantly more effective in the encryption quality of bitmap images than SHC in the case of images with large single colour areas and slightly more effective otherwise.

Keywords: Hill cipher, matrix, generalized permutation matrix, dynamic key, key generation, image encryption.

1 Introduction

The Hill cipher (HC) [1], [2] is computationally attractive as using multiplication of an mxm -sized key matrix, with entries from $Z_N = \{0,1,..,N-1\}$, by an *m*-component plaintext vector, with entries also from Z_N , to get a ciphertext vector, but it is vulnerable to the known plaintext-ciphertext attack (KPCA) [4] because of the use of the static key matrix. The Secure Hill cipher, SHC, [3] uses a dynamic key matrix obtained by random permutation of rows and columns of the master key matrix for each new plaintext. It transfers the HC-encrypted permutation to the receiving side together with the ciphertext. Thus, in SHC, each plaintext vector is encrypted by a new key matrix that prevents the KPCA on the vectors. But the permutations are transferred HC-encrypted, and the master key matrix can be revealed by the KPCA on the permutations [5]. A modification of SHC, SHC-M, [6] works as SHC does but without the permutations transfer. Instead, both sides use a pseudo-random permutation generator, and only the number of the necessary permutation is transferred to the receiver. It is shown in [7], [8] that both SHC and SHC-M fail to hide an image if it has large same colored areas.

Another HC modification [5], HCM-H, also uses dynamic key matrix produced with the help of a one way hash function applied to an integer picked up randomly by the sender to get the key matrix, and a vector added to the product of the key matrix with a plain text. It is shown in [7], [8] that HCM-H fails to hide an image if it has large same colored areas and it is vulnerable [9] to chosen-ciphertext attack because the selected random number is transmitted in clear over the communication link and is repeated.

We propose the modification of SHC/SHC-M denoted as SHC-GPM. The SHC-GPM uses a generalized permutation matrix (GPM), M_t , each row and column of which has all zero entries excepting only one entry co-prime to *N*, i.e.

$$\begin{aligned} &if\ \ j \neq t(i)\, then\, M_t(i,j) = 0, \\ &if\ \ j = t(i)\, then\, \gcd(M_t(i,j), N) = 1,\ i,j = \overline{1,m}, \end{aligned} \tag{1}$$

where *gcd* denotes the greatest common divisor, t is a secret permutation over Z_m , and $t(i)$ is an element of the permutation t at position *i*. To get a new GPM for each new plaintext vector, before encrypting, each non-zero entry of M_t is recalculated. The new key matrix in SHC-GPM is obtained by quite the same procedure as in SHC/SHC-M but we use the GPM instead of the permutation matrix. Entries of GPM are recalculated in a way allowing generating of a very large number of key matrices without repetition. Security of the proposed cipher is significantly better than the security of SHC/SHC-M because the number of non-repeatedly generated by SHC-GPM dynamic key matrices (number of dynamic keys) is significantly greater than that of SHC/SHC-M (e.g., 2^{48} times greater for *m=8, N=256*). Also, our experiments show that SHC-GPM is more efficient in image encryption quality than SHC-M.

The rest of the paper is organized as follows. Section 2 introduces SHC and SHC-M. Section 3 presents the proposed cipher. Section 4 introduces the performance and encryption quality of the proposed cipher. Security and statistical analysis of the proposed SHC-GPM are discussed in Section 5. A conclusion is given in Section 6. In the Appendix, an example of encryption and decryption by SHC-GPM is given together with the example of the images encrypted by SHC-GPM and AES.

2 Overview of SHC and SHC-M

Before beginning the overview of SHC and SHC-M, we define some notations which will be used throughout this paper: T_{Mul} , T_{Div} , T_{Add} and T_{Sub} , is the time for the scalar modular multiplication, division, addition and subtraction, respectively, T_{Move} is the time to move or assign a value, T_{PRPG} is the time for pseudo-random permutation generator, T_{Xor} is the time for xor-operation (exclusive OR). Note that the number of clock cycles as an example for Intel 80486 to perform multiplication and division is 18 while it is 1 for addition, subtraction, move, and exclusive OR operation [10]. It is worth noting that all matrices considered throughout the paper are *m x m* sized with entries over $Z_N = \{0,1,..,N-1\}$, hence all the operations in encryption/decryption algorithms are assumed *mod N*, where *m* (block size) and *N* (alphabet cardinality) are selected positive integers (e.g., *N=256* for gray scale images). Also, we assume that two parties, *A* and *B*, want to communicate securely, and *A* is a sender, and *B* is a receiver.

The SHC is a modification of HC. First, we introduce HC, SHC and then we describe SHC-M. Suppose two parties, a sender, *A*, and a receiver, *B*, want to exchange data using HC; they share an invertible key matrix *K*. If *A* wants to encrypt a plaintext vector, *P*, he gets the ciphertext vector, *C*, as follows:

$$C = KP \tag{2}$$

The receiver, *B*, decrypts *C* by

$$P = K^{-1}C \tag{3}$$

where K^{-1} is the key matrix inverse. The SHC differs from (2), (3) in the following. To encrypt a plaintext *P*, *A* selects a permutation, *t*, randomly over Z_m , builds a permutation matrix M_t , according to (1) (but with non-zero entries equal to one only), and gets K_t by permuting the rows and columns of a key matrix K

$$K_t = M_t \, K \, M_t^{-1} \tag{4}$$

The SHC encryption is then performed by (2), but using K_t instead of K . Additionally, sender *A* encrypts t by (2) using K and getting u as a ciphertext, and sends *C* and u together to the receiver. In order to decrypt the ciphertext, *B* decrypts t from u by (3), gets $(K^{-1})_t = (K_t)^{-1}$ from K^{-1} [3], and then reveals the plaintext by (3), using $(K^{-1})_t$ instead of K^{-1}. The number of dynamic keys used in SHC, *NDK(SHC)*, is

$$NDK(SHC) = m! \tag{5}$$

The computation complexity of SHC, *CC(SHC)*, can be calculated as follows. To obtain the K_t (4) we need to perform permutations of the rows and columns. Usually, to perform the permutation in one row we need *m* moves, therefore m^2 moves are required for *m* rows; similarly m^2 moves are required to perform the permutation over columns. Hence, $2m^2$ moves are required. To obtain the ciphertext (2), we multiply a matrix with a vector which needs $m^2 - m$ additions and m^2 multiplications; similarly to encrypt the permutation t we need $m^2 - m$ additions and m^2 multiplications, thus,

$$CC(SHC) = 2m^2 T_{Mul} + 2(m^2 - m)T_{Add} + 2m^2 T_{Move} \tag{6}$$

The SHC-M [6] uses the same initialization and the same encryption/decryption technique as SHC. But SHC-M assumes that the sender, *A*, and the receiver, *B*, share a secret seed value, *SEED*, which is used to generate a pseudo-random sequence of permutations. In order to encrypt a plaintext, the sender, *A*, selects a number *k*, and calculates

$$t = PRPermutationG(SEED, k, m) \tag{7}$$

getting the *k*-th output permutation from the pseudo-random permutation generator (*k* can be a block number in the sequence of transmitted blocks, or its function and *m* is the length of the permutation). Sender *A* then gets a ciphertext *C* as in SHC, and sends to receiver *B* both *C* and *k*.
In order to decrypt, *B* calculates t according to (7), and then gets the plaintext as in SHC. The number of dynamic keys used in SHC-M, *NDK(SHC-M)*, is given by (5) and the computation complexity of SHC-M, *CC(SHC-M)*, differs from (6) in the following. To generate the pseudo-random permutation (7) T_{PRPG} is required; additionally SHC-M does not encrypt the permutation t, therefore

$$CC(SHC-M) = m^2 T_{Mul} + (m^2 - m)T_{Add} + 2m^2 T_{Move} + T_{PRPG} \tag{8}$$

We can assess T_{PRPG} by consideration of RC4 [4] as a generator in SHC-M. The RC4 has two phases: the key setup and pseudorandom stream generator. We neglect the key setup phase because it is used only once in SHC-M for many permutations generated. In order to generate one output number, RC4 uses a state vector which is a permutation that is changed by swapping its two elements to produce one output number. To have significantly differing state vector permutation, we consider it as the output of PRPG after m RC4 iterations each of which requires 3 T_{Add}, 7 T_{Move}, and 3 $T_{M\,od}$, where note that $T_{M\,od} = T_{Mul} + T_{Div} + T_{Sub} = 2T_{Mul} + T_{Add}$ is the time for remainder after division calculation. Hence,

$$T_{PRPG} = m(6T_{Mul} + 6T_{Add} + 7T_{Move}) \tag{9}$$

3 THE PROPOSED SCH-GPM

The proposed cipher has the same structure, initialization, and the same encryption/decryption techniques as those of SHC and SHC-M, but it differs from SHC/SHC-M in the following. To start with, *A* calculates the permutation t according to (7) and constructs a GPM, M_t^0 according to (1) but with non-zero entries now not equal to 1. We assume $N = 2^n$ (in our experiments, *n=8*), and use the numbers of Z_N co-prime to N, Z_N^*, having the maximal order as non-zero entries of M_t^0; we keep them also in a vector $\mathrm{Re}dM_t^0$ such that $\mathrm{Re}dM_t^i(k) = M_t^i(k, t(k)) \in Z_{2^n}^*, i \geq 0, k = \overline{1,m}$

It is shown in [11] that the maximal order of elements in $Z_{2^n}^*$ is

$$mo = 2^{n-2}. \tag{10}$$

Also, we use a state vector, *St*, initialized by *m* zeroes, for key matrix modification; it works as a counter and may be viewed as an *m*-digit number base mo: $St_j \in Z_{mo}$ is it's j-th digit, $j = \overline{0, m-1}$. The plaintext P_i is encrypted using the key matrix

$$K_i = M_t^{i-1} K_{i-1} (M_t^{i-1})^{-1}. \tag{11}$$

as follows

$$C_i = K_i P_i \oplus \mathrm{Red}M_t^{i-1}, \tag{12}$$

where $\oplus$ is XOR, $i > 0$, $K_0 = K$, and

$$M_t^i(k,j) = \begin{cases} 0, j \neq t(k) \\ M_t^{i-1}(k,j)\cdot(\mathrm{Red}M_t^0(k))^s, j = t(k) \end{cases} \qquad j,k = \overline{1,m} \tag{13}$$

where

$$s = \begin{cases} 1, if\ St_j \neq (St+1)_j \\ 2, if\ St_j = (St+1)_j \end{cases}, \tag{14}$$

and the new value of the state vector is calculated

$$St = St + 1. \tag{15}$$

The numbers in (14) and (15) are added in the base mo (10). If the new value of *St* obtained in (15) is equal to zero, then a new permutation *t* is generated by (7): *A* selects a number of a permutation, generates it, and transfers the permutation's number to the receiver *B* so that he is able to generate the permutation by (7) as well, and all the steps (11)-(15) above are repeated.

The communicating parties share invertible key matrices K and M_t^0. The sender *A* sends to the receiver *B* the ciphertext C_i and the block number, i. Knowing i, *B* is able to calculate the key matrix K_i using (11), (13)-(15), and retrieve the plaintext as

$$P_i = K_i^{-1}(C_i \oplus \mathrm{Red}M_t^{i-1}). \tag{16}$$

Next key matrix inverse may be obtained from the previous key matrix inverse by (11), with key matrices replaced by their inverses. The computation complexity of SHC-GPM, *CC(SHC-GPM)*, both for encryption and for decryption, can be computed as follows.

According to (13)-(14), to obtain one modified element in $M_t^{(i)}$, at most two multiplications are required, therefore the time to modify *m* numbers is at most $2mT_{Mul}$, where $i > 0$. Additionally, the first part of (11) $X_i = M_t^{i-1}K_{i-1}$, can be performed in m^2T_{Mul} time due to the structure of M_t^{i-1}; similarly m^2T_{Mul} time is required to perform the second part of (11), $K_i = X_i(M_t^{i-1})^{-1}$. Inverse is a transposition of the original matrix with elements replaced by their inverses. Inverses of elements may be found using lookup tables, and, hence time of finding the matrix inverse is mT_{Move}. In addition, to perform (12), we need time $m^2T_{Mul} + (m^2 - m)T_{Add} + mT_{xor}$. Also a new permutation will be generated once for each set of mo^m encryptions by (7), it may be neglected, and time mT_{Add} is used in the worst case for (15). Hence,

$$CC(SHC-GPM) = (3m^2 + 2m)T_{Mul} + (m^2 - m)T_{Add} + mT_{xor} + mT_{Move}, \tag{17}$$

From (8), (9), and (17), we have approximately that

$$\frac{\mathrm{CC(SHC-GPM)}}{\mathrm{CC(SHC-M)}} \approx 3, \tag{18}$$

Since $3m^2T_{Mul}$ is the heaviest term in (17) and m^2T_{Mul} in (8)-(9). For the case of used in our experiments *m*=8 this ratio is

$$\frac{CC(SHC-GPM)}{CC(SHC-M)} = \frac{(3\cdot 8^2 + 2\cdot 8)\cdot 18 + (8^2 - 8)\cdot 1 + 8\cdot 1 + 8\cdot 1)}{8^2\cdot 18 + (8^2 - 8)\cdot 1 + 2\cdot 8^2\cdot 1 + 8(6\cdot 18 + 6\cdot 1 + 7\cdot 1)} = \frac{(208\cdot 18 + 80)}{(64\cdot 18 + 184 + 8\cdot 121)} = \frac{3824}{2034} \approx 2. \tag{19}$$

The number of dynamic keys of SHC-GPM is

$$NDK(SHC-GPM) = m! \cdot mo^{m}\ . \tag{20}$$

In the case of *N=256, m=8,*

$$NDK(SHC-GPM) = 8! \cdot (2^{8-2})^{8} = 8! \cdot 2^{48}\ . \tag{21}$$

$$NDK(SHC) = 8! \tag{22}$$

according to (5), (10) and (20). From (21), (22), one gets

$$\frac{NDK(SHC-GPM)}{NDK(SHC)} = 2^{48} \tag{23}$$

for *N=256, m=8*, hence security of the proposed algorithm is substantially better than that of SHC.

4 SHC-GPM Performance and Encryption Quality versus SHC-M

We developed programs for simulating SHC/SHC-M and SHC-GPM in C# with a Pentium(R) Duo (3.0 GHz) processor with 1-GB RAM on Windows XP. In our experiments, several RGB images are encrypted. Firstly, the image, P , of size NxM is converted into its RGB components. Afterwards, each colour matrix (R, G, B) is converted into a vector of integers within $\{0,1,...,255\}$. Each vector has the length $L = NxM$. Then, the so obtained three vectors represent the plaintext $P(3\times L)$ which will be encrypted using the block size *m*=8.

We examine the encryption quality for three different images containing very large single colour areas: Blackbox.bmp (Fig. 1), Nike.bmp (Fig. 2), and Symbol.bmp (Fig. 3). Also we examined the encryption quality for an image that does not contain many high frequency components: Lena.bmp (Fig. 4). The Girl.bmp (Fig. 5) is used as an example of an image containing many high frequency components. Each image is encrypted using SHC/SHC-M and SHC-GPM . The quality of encryption of these images is studied by visual inspection (Fig. 1, Fig. 2, Fig. 3, Fig.4, Fig. 5) and quantitavely (Table 1, used irregular deviation based quality measure ID [12], [13], [14]). This quality measure is calculated as follows:

1. Calculate the matrix, D, which represents the absolute value of the difference between each pixel value of the original and the encrypted image respectively:

   D = |O - E|,

   where O is the original (input) image and E is the encrypted (output) image.

2. Construct a histogram distribution of the D we get from step 1:

   h=histogram (D)

with 256 levels.

3. Get the average value of how many pixels are deviated at every deviation value by:

$$DC = \frac{1}{256}\sum_{i=0}^{255} h_i\ ,$$

4. Subtract this average from the deviation histogram and take the absolute value by:

$$AC(i) = |h_i - DC|,\ i=0,..,255.$$

5. Count:

$$ID = \sum_{i=0}^{255} AC(i).$$

The smaller ID, the better.

Based on visual inspection (Fig. 1, Fig. 2, Fig. 3), it is obvious that the proposed scheme SHC-GPM has better encryption quality than the SHC/SHC-M. The SHC/SHC-M fails to hide the plain-image of large single color area, especially for the black color represented by zero value for both plaintext/ciphertext obtained by SHC/SHC-M but SHC-GPM succeeds in hiding all the features of the image containing large single colour areas (Fig. 1, Fig. 2, Fig. 3). On the other hand from the numerical evaluation of encryption quality measure ID (Table 1), we note that the proposed scheme SHC-GPM versus SHC/SHC-M give better encryption quality. SHC-GPM, and SHC/SHC-M are all good in encrypting images containing many high frequency components; SHC-GPM and SHC/SHC-M give nearly the same results but the SHC-GPM is more effective (Table 1, rows 4-5).

Image/Algorithm	SHC/SHC-M	SHC-GPM
Blackbox.bmp	34036.281	8960.135
Nike.bmp	23980.791	13455.083
Symbol.bmp	10482.25	4289.96
Lena.bmp	10256	10238.67
Girl.bmp	11459.55	9791.92

Table 1 The numerical evaluations of irregular deviation ID for encrypted images with SHC/SHC-M and SHC-GPM; the smaller ID, the better.

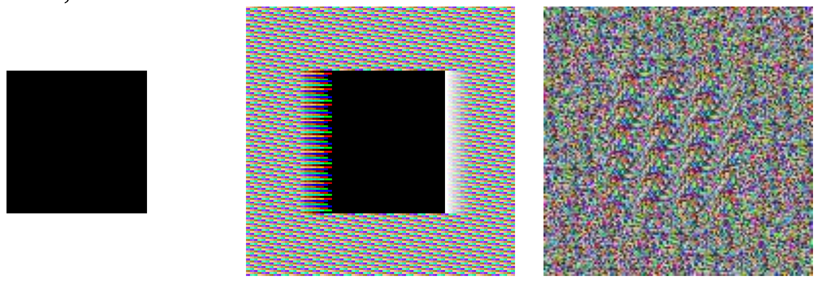

(a) (b) (b)

Figure 1 a) Original Blackbox.bmp encrypted by: b) SHC/SHC-M, c) SHC-GPM

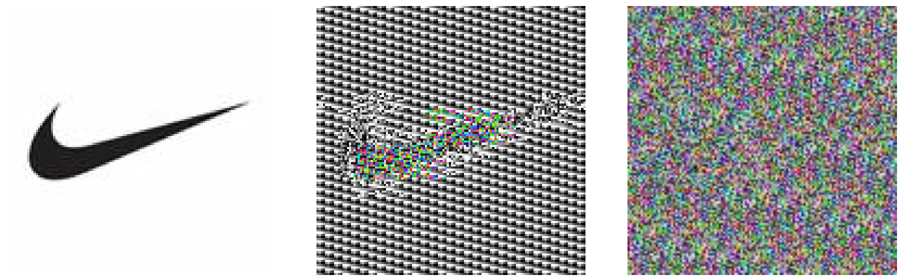

(a) (b) (c)

Figure 2 a) Original Nike.bmp encrypted by: b) SHC/SHC-M, c) SHC-GPM

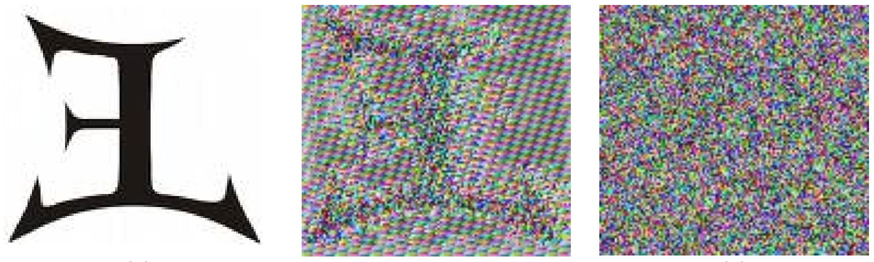
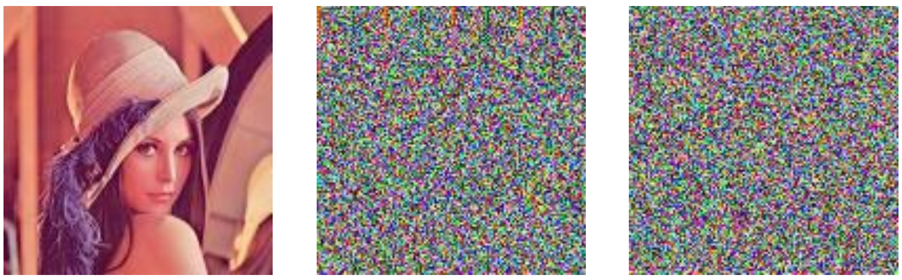

(a) (b) (c)

Figure 3 a) Original Symbol.bmp encrypted by: b) SHC/SHC-M, c) SHC-GPM



(a) (b) (c)

Figure 4 a) Original Lena.bmp encrypted by: b) SHC/SHC-M, c) SHC-GPM

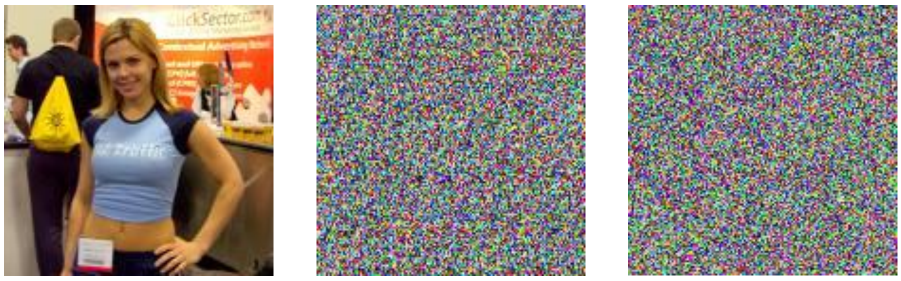

(a) (b) (c)

Figure 5 a) Original Girl.bmp encrypted by: b) SHC/SHC-M, c) SHC-GPM

We examined the encryption time for the Blackbox.bmp image having 138 × 138 pixels and 56.1 KB size. The encryption time measured when applying SHC-GPM is 137 ms, while it is 59 ms when applying SHC-M that complies with (19). Based on the encryption time, and the obtained number of clock cycles according to (8), (9), (17), it is clear that SHC-M roughly is two-three times better than SHC-GPM in the encryption time, but (5), and (20)-(23) show that *NDK(SHC-GPM)* is significantly greater than *NDK(SHC-M)*, therefore SHC-GPM is more secure.

5 Security and Statistical Analysis

The security of a cryptosystem is determined by the ability to resist all kinds of cryptanalysis and attacks [15], [16], [17], [18]. Robustness against attacks is used to evaluate the security of our scheme. The results show the satisfactory security of the SHC-GPM as explained and discussed in the following subsections.

5.1. Key Space Analysis

Key space is known as the total number of different keys that can be used in encryption. For a good encryption algorithm, it is necessary to have a large enough key space to make brute force attack infeasible. For the SHC-GPM, the key space is the same as that of HC [3], [19]. Therefore the key space of the scheme is large; hence it is secure against brute force attack.

5.2. Known Plaintext-Ciphertext Attack

The KPCA is effective if a same key is used to encrypt many plaintexts. Similar to SHC [3], our proposed scheme SCH-GPM is secure against the KPCA since each plaintext is encrypted by a different key, and the number of such dynamic keys is significantly large (20). Equations (5) and (20) show that the NDK(SHC-GPM) (20) is larger than the NDK(SHC) (5); hence SHC-GPM is more secure.

5.3. Statistical Analysis Resistance

A good cipher should be robust against any statistical attack. If this proposed scheme can confuse images to the one with random distribution, then it is difficult for statistical attackers. To prove the robustness of the proposed scheme, the statistical analysis has been performed. It is usually evaluated by the following measures [15], [17], [20], [21], [22], [23]: calculating the histograms of the encrypted images and the correlation of two adjacent pixels in the plain/encrypted image demonstrating their superior confusion and diffusion property. The obtained results show that SHC-GPM strongly withstands statistical attacks.

5.3.1. Histograms of encrypted images

We have computed and analyzed the histogram of the encrypted image as well as its original image; a typical example is given in Fig. 6. The histogram of the encrypted image is very close to uniform distribution; it is significantly different from the original image, and bears no statistical resemblance to the original image.

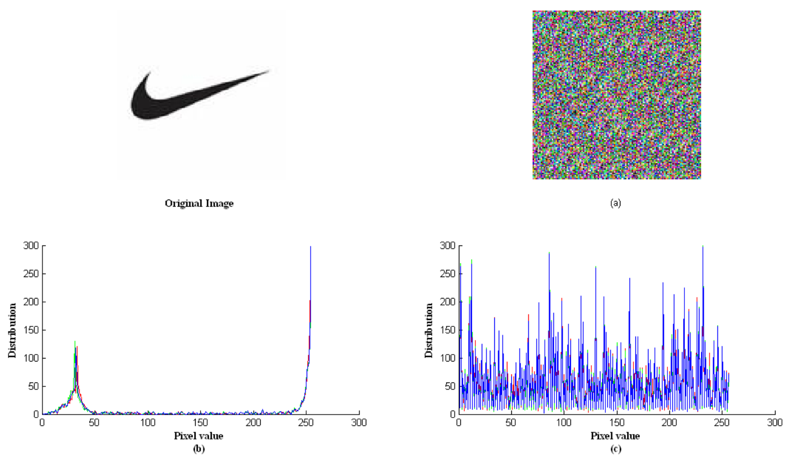


Figure 6 Histogram of RGB layers for original/encrypted Nike.bmp: a) SHC-GPM-encrypted, b) histogram of the original image, c) histogram of SHC-GPM-encrypted.

5.3.2. Correlation of Two Adjacent Pixels

There is a very good correlation between adjacent pixels in the plain-image. We studied the correlation between two adjacent pixels in plain-image and encrypted image in three different orientations (horizontal, vertical and diagonal). We use the following procedure: first 1000 pairs of two adjacent pixels in three different orientations are selected randomly from image to test correlation, and then we calculate the correlation coefficient C.C (explained in the Appendix) of each pair.

Table 2 shows the numerical evaluation of the calculated correlation coefficients of two adjacent pixels in Nike.bmp encrypted by SHC/SHC-M and SHC-GPM in three different orientations as a practical example. It is clear that, the neighboring pixels in the plain-image have a very high correlation while they have a very small correlation (the closer to zero, the better) for encrypted image. This proves that the proposed encryption scheme SHC-GPM satisfies very small correlation and is better than SHC.

Image	Direction	Plain Image	Encrypted Image	
			SHC/SHC-M	SHC-GPM
Nike.bmp	Horizontal	0.9413	0.0849	0.0583
	Vertical	0.9031	0.1484	0.0098
	Diagonal	0.9801	0.5743	0.0140

Table 2 Correlation coefficients of two adjacent pixels in original and SHC/SHC-M-encrypted images and SHC-GPM-encrypted images.

6 Conclusion

A new cipher, SHC-GPM, is presented that is a modification of SHC/SHC-M. In this paper, SHC/SHC-M and the proposed SHC-GPM have been implemented for image encryption. Quality of image encryption is studied using visual inspection and numerical quality measures. From the obtained results, it follows that the proposed SHC-GPM is more effective in encryption quality than SHC/SHC-M especially for images having substantial same color areas (see Fig. 1, Fig. 2, Fig. 3, Table 1). Versus SHC/SHC-M, the proposed cipher SHC-GPM significantly increases the security, based on the greater number of dynamic keys used (see (23)). These improvements are got by expense of two-three times decreasing performance because SHC-GPM uses generalized permutations matrices which need approximately three times greater number of multiplications compared to the permutations used by SHC/SHC-M. An important feature of the proposed cipher is the use of numbers having known in advance maximal order (10) allowing guaranteeing non-repeating sequence of the key matrices of any length by respective choice of the parameters (for parameter values used in the reported experiments, the length (*NDK*) is given by (21)). Proposed for SHC-

GPM method of the sequence of keys generation (represented by (7), (10), (11), and (13)-(15)) might be used also as a basis for a cryptographically strong pseudo-random number generator with guaranteed very large period. SHC-GPM resists the KPCA because of the use of dynamically changing key matrices similar to SHC but the proposed SHC-GPM is more secure than SHC because of the significantly larger number of dynamic keys generated: (20) versus (5). Experimental analysis also shows that the SHC-GPM resists the statistical attacks.

APPENDIX

Correlation Based Quality Measure

A good encryption algorithm must produce encrypted image totally random patterns hiding all the features of the original image; the encrypted image must be independent of the original image. This means that the two images must have a correlation coefficient very close to zero. The correlation coefficient is given by the following expression:

$$C.C=\frac{\sum_{i=1}^{N}(x_i-E(x))(y_i-E(y))}{\sqrt{\sum_{i=1}^{N}(x_i-E(x))^2}\sqrt{\sum_{i=1}^{N}(y_i-E(y))^2}}.$$

where $E(x)=\frac{1}{N}\sum_{i=1}^{N}x_i$, x and y are grayscale values of two adjacent pixels of the original and encrypted images. Note that: the closer C.C to zero, the better.

SHC-GPM Encryption and Decryption Example

Let $N=256$, $m=2$, $St=[0,0]$, $t=[2,1]$, $M_t^0=\begin{bmatrix}0 & 43\\37 & 0\end{bmatrix}$, $(M_t^0)^{-1}=\begin{bmatrix}0 & 173\\131 & 0\end{bmatrix}$, $\mathrm{Red}M_t^0=\begin{bmatrix}43\\37\end{bmatrix}$, $K=\begin{bmatrix}14 & 95\\3 & 27\end{bmatrix}$, and the plaintext to be encrypted is "*251, 241, 13, 25, 28, 31*". Note that by (10), $mo=2^{8-2}=64$ in the case of $N=256$, $\det(K)=93 \bmod 256$, and $K^{-1}=245\times\begin{bmatrix}27 & -95\\-3 & 14\end{bmatrix}=\begin{bmatrix}215 & 21\\33 & 102\end{bmatrix}$.

First block key and encryption:

$$K_1=\begin{bmatrix}0 & 43\\37 & 0\end{bmatrix}\begin{bmatrix}14 & 95\\3 & 27\end{bmatrix}\begin{bmatrix}0 & 37^{-1}\\43^{-1} & 0\end{bmatrix}=\begin{bmatrix}129 & 137\\6 & 187\end{bmatrix}\begin{bmatrix}0 & 173\\131 & 0\end{bmatrix}=\begin{bmatrix}27 & 45\\177 & 14\end{bmatrix}$$, then

$$C_1=\begin{bmatrix}27 & 45\\177 & 14\end{bmatrix}\begin{bmatrix}251\\241\end{bmatrix}\oplus\begin{bmatrix}43\\37\end{bmatrix}=\begin{bmatrix}214\\185\end{bmatrix}\oplus\begin{bmatrix}43\\37\end{bmatrix}=\begin{bmatrix}253\\156\end{bmatrix}$$

Second block key and encryption:

$St=[0,1]$ by (15) and according to (13)-(14) $M_t^1(1,2)=M_t^0(1,2)\cdot(\mathrm{Red}M_t^0(1))^2=43^3=147$,

$M_t^1(2,1)=M_t^0(2,1)\cdot(\mathrm{Red}M_t^0(2))^1=37^2=89$

$$K_2=\begin{bmatrix}0 & 147\\89 & 0\end{bmatrix}\begin{bmatrix}27 & 45\\177 & 14\end{bmatrix}\begin{bmatrix}0 & 89^{-1}\\147^{-1} & 0\end{bmatrix}=\begin{bmatrix}163 & 10\\99 & 165\end{bmatrix}\begin{bmatrix}0 & 233\\155 & 0\end{bmatrix}=\begin{bmatrix}14 & 91\\231 & 27\end{bmatrix},$$

then $C_2=\begin{bmatrix}14 & 91\\231 & 27\end{bmatrix}\begin{bmatrix}13\\25\end{bmatrix}\oplus\begin{bmatrix}147\\89\end{bmatrix}=\begin{bmatrix}153\\94\end{bmatrix}\oplus\begin{bmatrix}147\\89\end{bmatrix}=\begin{bmatrix}10\\7\end{bmatrix}$

Third block key and encryption:

$St=[0,2]$ by (15) and according to (13)-(14)

$M_t^2(1,2)=M_t^1(1,2)\cdot(\mathrm{Red}M_t^0(1))^2=147\cdot 43^2=187$, $M_t^2(2,1)=M_t^1(2,1)\cdot(\mathrm{Red}M_t^0(2))^1=89\cdot 37=221$

$$K_3 = \begin{bmatrix} 0 & 187 \\ 221 & 0 \end{bmatrix}\begin{bmatrix} 14 & 91 \\ 231 & 27 \end{bmatrix}\begin{bmatrix} 0 & 221^{-1} \\ 187^{-1} & 0 \end{bmatrix} = \begin{bmatrix} 189 & 185 \\ 22 & 143 \end{bmatrix}\begin{bmatrix} 0 & 117 \\ 115 & 0 \end{bmatrix} = \begin{bmatrix} 27 & 97 \\ 61 & 14 \end{bmatrix},$$

then $C_3 = \begin{bmatrix} 27 & 97 \\ 61 & 14 \end{bmatrix}\begin{bmatrix} 28 \\ 31 \end{bmatrix} \oplus \begin{bmatrix} 187 \\ 221 \end{bmatrix} = \begin{bmatrix} 179 \\ 94 \end{bmatrix} \oplus \begin{bmatrix} 187 \\ 221 \end{bmatrix} = \begin{bmatrix} 8 \\ 131 \end{bmatrix}.$

Thus far, the ciphertext is "*253, 156, 10, 7, 8, 131*".

The decryption process is depicted as follows. Since the receiver has the key matrices K , M_t^0 and the encrypted block number, the receiver can calculate the inverse of the key matrix K_i^{-1} using (11), (13)-(15) as in encryption process, and use (16) to retrieve the plaintext.

First key matrix inverse according to (11):

$K_1^{-1} = \begin{bmatrix} 0 & 43 \\ 37 & 0 \end{bmatrix}\begin{bmatrix} 215 & 21 \\ 33 & 102 \end{bmatrix}\begin{bmatrix} 0 & 173 \\ 131 & 0 \end{bmatrix} = \begin{bmatrix} 102 & 239 \\ 155 & 215 \end{bmatrix}$. Then, according to (16)

$$P_1 = \begin{bmatrix} 102 & 239 \\ 155 & 215 \end{bmatrix}\left(\begin{bmatrix} 253 \\ 156 \end{bmatrix} \oplus \begin{bmatrix} 43 \\ 37 \end{bmatrix}\right) = \begin{bmatrix} 102 & 239 \\ 155 & 215 \end{bmatrix}\begin{bmatrix} 214 \\ 185 \end{bmatrix} = \begin{bmatrix} 251 \\ 241 \end{bmatrix}.$$

Second block key and decryption:

$$K_2^{-1} = \begin{bmatrix} 0 & 147 \\ 89 & 0 \end{bmatrix}\begin{bmatrix} 102 & 239 \\ 155 & 219 \end{bmatrix}\begin{bmatrix} 0 & 233 \\ 155 & 0 \end{bmatrix} = \begin{bmatrix} 215 & 233 \\ 237 & 102 \end{bmatrix}.$$

Then according to (16) $P_2 = \begin{bmatrix} 215 & 233 \\ 237 & 102 \end{bmatrix}\left(\begin{bmatrix} 10 \\ 7 \end{bmatrix} \oplus \begin{bmatrix} 147 \\ 89 \end{bmatrix}\right) = \begin{bmatrix} 215 & 233 \\ 237 & 102 \end{bmatrix}\begin{bmatrix} 153 \\ 94 \end{bmatrix} = \begin{bmatrix} 13 \\ 25 \end{bmatrix}$

Third block key and decryption:

$K_3^{-1} = \begin{bmatrix} 0 & 187 \\ 221 & 0 \end{bmatrix}\begin{bmatrix} 215 & 233 \\ 237 & 102 \end{bmatrix}\begin{bmatrix} 0 & 117 \\ 115 & 0 \end{bmatrix} = \begin{bmatrix} 102 & 43 \\ 159 & 215 \end{bmatrix}$. Then according to (16)

$$P_3 = \begin{bmatrix} 102 & 43 \\ 159 & 215 \end{bmatrix}\left(\begin{bmatrix} 8 \\ 131 \end{bmatrix} \oplus \begin{bmatrix} 187 \\ 221 \end{bmatrix}\right) = \begin{bmatrix} 102 & 43 \\ 159 & 215 \end{bmatrix}\begin{bmatrix} 179 \\ 94 \end{bmatrix} = \begin{bmatrix} 28 \\ 31 \end{bmatrix}.$$

Thus far, the plaintext revealed is "*251, 241, 13, 25, 28, 31*" that is equal to the original one.

SHC-GPM versus AES

To give adequate performance comparison, we examine our proposed SHC-GPM versus other well known algorithms (e.g. AES). We examined the encryption quality of several images. Based on visual inspection, the proposed SHC-GPM encrypts the images with large single colour areas (identical plaintext blocks), it successfully hides data patterns. The AES fails to hide the data patterns for the images contain large single colour areas (Mecky.bmp: Fig. 8, Penguin.bmp: Fig. 9, and bicycle.bmp: Fig. 10). That is, the proposed SHC-GPM has advantage in encryption of identical plaintext blocks over the AES.

Image/Algorithm	SHC-GPM	AES
Mecy.bmp	12872.58	47726.75
bicycle.bmp	9736.79	25031.32
Penguin .bmp	6440.06	20745.34

Table 3 ID for encrypted images using SHC-GPM and AES, *m=16.*

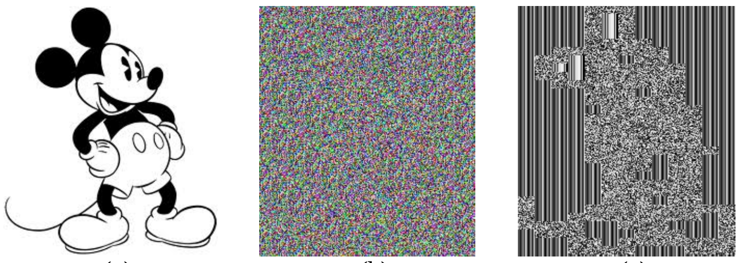

(a) (b) (c)

Figure 7 a) Mecky.bmp encrypted by: b) SHC-GPM, c) AES

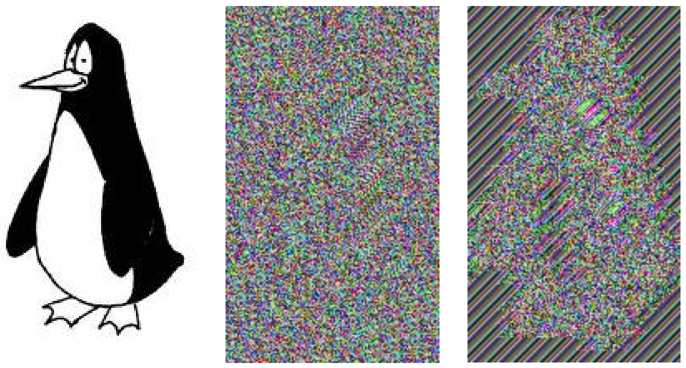


Figure 8 a) Penguin.bmp encrypted by: b) SHC-GPM, c) AES.

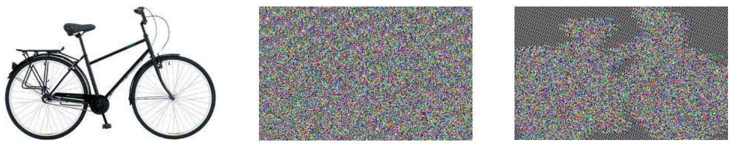


Figure 9 a) Bicycle.bmp encrypted by: b) SHC-GPM, c) AES.

References

[1] L. S. Hill, *Cryptography in an Algebraic Alphabet*, American Mathematical Monthly, 36(6):306-312 (1929).

[2] L. S. Hill *Concerning Certain Linear Transformation Apparatus of Cryptography*, American Mathematical Monthly, 38(3): 135-154 (1931).

[3] S. Saeednia *How to Make the Hill Cipher Secure*, Journal of Cryptologia, 24(4): 353-360 (2000).

[4] W. Stallings, *Cryptography and Network Security Principles and Practices* (4th edn.). Prentice Hall: New Jersey, (2006).

[5] C.H. Lin, C.Y. Lee, *Comments on Saeednia's Improved Scheme for the Hill Cipher*, Journal of the Chinese Institute of Engineers, 27(5): 743-746 (2004).

[6] A. Chefranov, *Secure Hill Cipher Modification SHC-M*, Proc. of the First International Conference on Security of Information and Networks (SIN2007), Gazimagusa (TRNC) North Cyprus, Elçi, A., Ors, B., and Preneel, B. (Eds.) Trafford Publishing, Canada. 34-37 (2007).

[7] A.Y. Mahmoud, A.G. Chefranov, *Hill Cipher Modification Based on Eigenvalues HCM-EE*. Proc. of the Second International Conference on Security of Information and Networks (SIN2009), Gazimagusa (TRNC) North Cyprus, Elci, A., Orgun, M., and Chefranov, A. (Eds.) ACM, New York, USA, 2009: 164- 167 (2009).

[8] A.Y. Mahmoud, A.G. Chefranov, *Hill Cipher Modification Based on Pseudo-Random Eigenvalues*. Fourth coming paper, to appear in Applied Mathematics & Information Sciences.

[9] T. Mohsen, F. Abolfazl, *A Secure Cryptosystem Based on Affine Transformation*. John Wiley & Sons, 4(2): 207-215 (2011).

[10] M.A. Mazidi, J.L. Mazidi, *The 80x86 IBM PC and Compatible Computers*. (volumes I & II, 3rd ed). Prentice Hall, (1995).

[11] M. A. Tom, *Introduction to Analytic Number Theory* (1st edn.). Springer-Verlag, (1976).

[12] H. Elkamchouchi, A.M. Makar, *Measuring Encryption Quality of Bitmaps Images with Rijndael and KAMKAR Block Ciphers*. Proc. Twenty Second National Radio Science Conference (NRSC), Egypt 1-8 (2005).

[13] A.I. Ismail, M. Amin, H. Diab, *How to Repair the Hill Cipher*. Journal of Zhejiang University Sci. A, 7(12): 2022-2030 (2006).

[14] I. Ziedan, M. Fouad, H.D. Salem, *Application of Data Encryption Standard to Bitmap and JPEG Images*. Proc. Twentieth National Radio Science Conference (NRSC), Egypt, 1-16 (2003).

[15] E.H. Hossam, H.M. Ahmed, S.F. Osama, *An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) for Image Encryption and Decryption*. Journal of Computing and Informatics, 31(1): 121-129 (2007).

[16] M.A. Yaobin, C.H. Guanrong, *Chaos-Based Image Encryption in Eduardo Bayro-Corrochano editor*, Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neural Computing and Robotics. Springer-Verlag,

Heidelberg, (2004).
[17] M.A. Yaobin, C.H. Guanrong, L. Shiguo, *A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps*. Chaos Solitons and Fractals, 21(3): 749-761 (2004).
[18] G. Alvarez, S. Li, *Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems*. International Journal of Bifurcation and Chaos, 16(8): 2129–2151 (2006).
[19] J. Overbey, W. Traves, J. Wojdylo. *On the Key Space of the Hill Cipher*. Journal of Cryptologia, 29(1): 59-72 (2005).
[20] E.H Hossam, H.M. Ahmed, S.F. Osama, *Encryption Efficiency Analysis and Security Evaluation of RC6 Block Ciphers for Digital Images*. International Journal of Computer and Information Technology, 1(1): 33-39 (2007).
[21] H. Cheng, L. Xiaobo, *Partial Encryption of Compressed Images and Videos*. IEEE Trans. Signal Process, 48 (8): 2439-2451 (2000).
[22] L.M. Marvel, G.G. Boncelet, C.T. Retter, *Spread Spectrum Image Steganography*. IEEE Trans. Signal Process, 8(8): 1075-1083(1999).
[23] S. Lian, J. Sun, Z. Wong, *Security Analysis of Chaos-Based Image Encryption Algorithm*. Phys Lett A 351, 645–661 (2005).