I.J. Computer Network and Information Security, **2014, 5, 57-65**
Published Online April 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2014.05.08

A Hill Cipher Modification Based on Eigenvalues Extension with Dynamic Key Size HCM-EXDKS

Ahmed Y. Mahmoud[1], Alexander G. Chefranov[2]
[1]Department of Information Technology, Al-Azhar University-Gaza, Gaza, Palestine
[2]Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus
[1]ahmed@alazhar.edu.ps, [2]alexander.chefranov@emu.edu.tr

Abstract—All the proposed Hill cipher modifications have been restricted to the use of dynamic keys only. In this paper, we propose an extension of Hill cipher modification based on eigenvalues HCM-EE, called HCM-EXDKS. The proposed extension generating dynamic encryption key matrix by exponentiation that is made efficiently with the help of eigenvalues, HCM-EXDKS introduces a new class of dynamic keys together with dynamically changing key size. Security of HCM-EXDKS is provided by the use of a large number of dynamic keys with variable size. The proposed extension is more effective in the encryption quality of RGB images than HCM-EE and Hill cipher-known modifications in the case of images with large single colour areas and slightly more effective otherwise. HCM-EXDKS almost has the same encryption time as HCM-EE, and HCM-HMAC. HCM-EXDKS is two times faster than HCM-H, having the best encryption quality among Hill cipher modifications compared versus HCM-EXDKS.

Index Terms—Hill cipher, eiqenvalue exponentiation, pseudorandom number, dynamic key, dynamic key size, image encryption.

I. Introduction

The Hill cipher (HC) is a well-known symmetric cryptosystem [1], [2]. The core of HC is matrix manipulations; it multiplies a plaintext vector by a key matrix to get the ciphertext. The HC is extremely secure against ciphertext only and brute force attacks. That is because the key space is extremely large, due to choosing the matrix elements from a large set of integers [3], it is also resistant to the frequency letter analysis, but it can be broken by the known plaintext-ciphertext attack (KPCA) [4], [5], [6], [7]. The key matrix can be calculated easily from a set of known plaintext-ciphertext pairs. The vulnerability of the HC to the KPCA makes it unusable in practice. Security of HC was improved in [5], [8], [9], [10]. One of proposed their methods, HCM-PT, uses a dynamic key matrix obtained by permutations of rows and columns from the master key matrix to get every next ciphertext, and transfers it together with an HC-encrypted permutation to the receiving side. Thus, in HCM-PT, each plaintext vector is encrypted by a new dynamic key matrix that prevents the KPCA; the number of possible dynamic keys is equal to the number of permutations of the key matrix rows, and it may be used as a characteristic of its security. But permutations in HCM-PT are transferred HC-encrypted, which means that master key matrix can be revealed by the KPCA on the transferred encrypted permutations [10]. Modification [8], HCM-NPT, works as HCM-PT does, but without permutations transfer; instead, both communicating parties use a pseudo-random permutation generator, and only the consecutive number of the necessary permutation is transferred to the receiver. It has good computational complexity and the number of its dynamic keys is the same as for HCM-PT, but [11] shows that HCM-NPT is not effective in the encryption quality of RGB bitmap images in the case of images with large single colour areas.

Another HC modification [9], HILLMRIV, also uses dynamic key matrices: it modifies each row of the matrix key by multiplying the current key by a secret initial vector. But HILLMRIV is still vulnerable to KPCA [12], [13]. Another HC modification [10], 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. HCM-H is vulnerable [14] to chosen-ciphertext attack because the selected random number is transmitted in clear over the communication link and is repeated. To avoid this random number transfer, a modification of HCM-H [14], HCM-HMAC, uses only a seed value secure transfer, and then both parties generate necessary numbers synchronously, where HMAC is a hash function, e.g., MD5[15], SHA-1[16]. The difference between HCM-H and HCM-HMAC is similar to the difference between HCM-PT and HCM-NPT. Despite these improvement, the Hill cipher still either susceptible to the KPCA or ineffective in image encryption in the case of images with large single colour area.

Up to now, HC modifications consider the dynamic keys as the major solution for enhancing the security and reducing the risk of cryptanalysis of the HC. However, using the dynamic keys is not always the best solution, since the generated keys are dependent on the initial parameters. In this paper, we propose an extension of the HC modification, HCM-EE [11]. HCM-EE generates dynamic encryption key matrix efficiently with the help of eigenvalues; it uses the eigenvalues for matrix exponentiation to a pseudo-random power for a new key matrix generating for each plaintext block. The proposed extension uses the dynamic keys together with dynamically changing key size instead of using dynamic keys only.

The paper is organized as follows. Section II briefly introduces the Hill cipher, HCM-PT, HCM-NPT, HCM-H, HCM-HMAC, and HCM-EE. Section III is devoted to the proposed extension HCM-EXDKS. In Section IV, comparison of the proposed extension, HCM-EXDKS, versus HC modifications is given; estimates of their execution time and experimental results on image encryption are presented. Conclusion is in the Section V.

II. Overview of the Hill Cipher and its Modifications

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.

First, we introduce HC, HCM-PT, HCM-NPT, HCM-H, HCM-HMAC and then we describe HCM-EE.

When HC is used, A and B share an invertible key matrix K. Sender A encrypts a plaintext vector, P:

$$C = K \cdot P \tag{1}$$

The receiver, *B*, decrypts the ciphertext vector *C* by

$$P = K^{-1} \cdot C, \tag{2}$$

where K^{-1} is the key inverse. For existence of K^{-1}, we require

$$gcd(det(K) \, mod \, N, N) = 1. \tag{3}$$

where *gcd* is the greatest common divisor and *det(K)* denotes the determinant of *K*.

HCM-PT [5] differs from HC in the following. To encrypt a plaintext P, A randomly selects a permutation, t, of Zm, and permutes the rows and columns of a key matrix K according to t producing a new key-matrix Kt=t(K). HCM-PT encryption is then performed by (1), but using Kt instead of K. Additionally, sender A encrypts t by (1) 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 (2), gets (K-1)t = (Kt)-1 [5] from K-1, and then reveals the plaintext by (2), using (K-1)t instead of K-1. The number of dynamic keys used in HCM-PT is

$$NDK(HCM\text{-}PT) = m!. \tag{4}$$

HCM-NPT [8] uses the same initialization and the same encryption/decryption technique as HCM-PT does. But HCM-NPT 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 r, and calculates.

$$t_r = PRPermutationG(SEED, r), \tag{5}$$

getting the *r*-th output permutation from the pseudo-random permutation generator *PRPermutationG* (*r* can be a block number in the sequence of transmitted blocks, or its function). Sender *A* then gets a ciphertext *C* as in HCM-PT, and sends to receiver *B* both *C* and *r*. In order to decrypt, *B* calculates t_r according to (5), and then gets the plaintext as in HCM-PT. The number of dynamic keys used in HCM-NPT, *NDK(HCM-NPT)*, is the same as *NDK(HCM-PT)* (4).

Proposed in [10], another HC modification, HCM-H, works as follows. The sender, A, and the receiver, B, share an invertible matrix K. To encrypt the plaintext P, A, selects a random integer a, where $0 < a < N$, and applies a one way hash function to compute the parameter $b = f\,(a \| k_{11} \| k_{12} \| ... \| k_{mm})$, where $k_{11}, k_{12}, ..., k_{mm}$ are the elements of K; b is used to select the k_{ij} from K, where i and j can be calculated according to (6)

$$i = \left\lfloor \frac{b-1}{m} \right\rfloor \cdot (\mathrm{mod}\, m) + 1, \; j = b - \left\lfloor \frac{b-1}{m} \right\rfloor \cdot m. \tag{6}$$

Then, A generates a vector $V = [v_1, v_2, ..., v_m]$ according to (7)

$$v_1 = f(k_{ij}) \bmod N, v_2 = f(v_1) \bmod N = f^2(k_{ij}) \bmod N, ..., v_m = f(v_{m-1}) \bmod N = f^m(k_{ij}) \bmod N. \tag{7}$$

Then, *A* encrypts the plaintext *P* by

$$C = k_{ij} \cdot P \cdot K + V, \tag{8}$$

and sends together *C* and *a* to *B*. The decryption process is done by

$$P = k_{ij}^{-1} \cdot (C - V) \cdot K^{-1}. \tag{9}$$

The number of dynamic keys used in HCM-H is

$$NDK(HCM\text{-}H) = min(m^2, N). \tag{10}$$

Proposed in [14], HCM-HMAC, works as follows. In order to transfer a seed value, the sender, A, transmits the seed value a according to the Hughes key-exchange protocol [17]. Then the seed value a_0 can be used to generate the chain of pseudo-random numbers synchronously by the both parties; a_t can be calculated by

$$a_t = HMAC_{k'}(a_{t-1}), t = 1, 2, \ldots, \tag{11}$$

where k' is the secret key of the hash function, k' can calculated by

$$k' = (k_{11} \| k_{12} \| k_{13} \| \ldots \| k_{mm} \| a_{t-1}) \bmod 2^q, \tag{12}$$

where || denotes the concatenation, q is the number of bits required for the hash function, and a_t is used in recursive calculations of the vector $V=[v_1, v_2, \ldots, v_n]$, calculated for the encryption of t-th block, $v_0 = 1$, if $a_t \equiv 0 (\bmod p)$ otherwise $v_0 = a_t \bmod p$, p is a prime number.

$$v_i = k_{ij} + \tilde{v}_{i-1} a_t \bmod p, \quad i = 1, 2, \ldots m, \quad \text{and} \quad j = (v_{i-1} \bmod m) + 1 \tag{13}$$

$\tilde{v}_{i-1}$ is calculated by

$$\tilde{v}_{i-1} = 2^{\left\lceil \frac{\gamma}{2} \right\rceil} + \left(v_{i-1} \bmod 2^{\left\lceil \frac{\gamma}{2} \right\rceil} \right), \ \gamma = \lfloor \log_2 v_{i-1} \rfloor + 1 \tag{14}$$

where $\gamma = \lfloor \log_2 v_{i-1} \rfloor + 1$ denotes the bit length of v_{i-1}. Then, A encrypts the plaintext P_t by

$$C_t = v_0 \cdot P_t \cdot K + V \bmod P, \tag{15}$$

and sends together C_t and a to B, t=1,2,... The receiver B calculates the required parameters by using (11)-(14), and then gets the plaintext by

$$P_t = v_0^{-1} \cdot (C_t - V) \cdot K^{-1} \bmod P. \tag{16}$$

HCM-EE [11] works as follows. Sender A selects a set $E = \{e_1, e_2, \ldots, e_m\} \subset Z_N - \{0\}$, gcd(ej, N)=1, gcd is the greatest common divisor, 1≤ j≤ m; at least one ej should have the maximal order which is $\frac{\varphi(N)}{2}$ for N being a power of 2 [18], $\varphi(N)$ is the Euler's totient function [4], giving the number of positive integers less than N and co-prime to it. Then A constructs an invertible matrix Q and calculates the key matrix K [19]:

$$K = Q \cdot D \cdot Q^{-1}, \tag{17}$$

where D is a diagonal matrix, diagonal elements of which are its eigenvalues from E. Note that Q and D satisfy (3); A and B share them securely. Additionally, they share the secret values, *SEEDl* and *SEEDt*; *SEEDl* is used to generate the set of pseudo-random numbers $l = \{l_1, l_2, \ldots, l_n\}$ by (18), $l_i \neq 0$ and $l_i \in \{2, \ldots, \varphi(N) - 1\}$, $1 \le i \le n$, n is the number of blocks. *SEEDt* is used to generate a pseudo-random sequence of permutations t. In order to encrypt the i-th plaintext block P_i, A selects

$$l_i = PRNG(SEEDl, i) > 0, \tag{18}$$

then calculates

$$E_i = \{e_j^{l_i}\}_{t_r}, 1 \le j \le m, 1 \le i \le n, \tag{19}$$

where $e_j \in E$, n is the number of blocks, and the random permutation t_r can be obtained by (5). Finally, A calculates

$$KM_i = Q \cdot D_i \cdot Q^{-1}, \tag{20}$$

where D_i is a diagonal matrix, diagonal elements of which are from E_i (19) after exponentiation to l_i and permutation t_r are performed and

$$i = \frac{\varphi(N)}{2} \cdot r + s, \ 0 \le s < \frac{\varphi(N)}{2}. \tag{21}$$

The plaintext P_i is encrypted as follows

$$C_i = KM_i \cdot P_i + diag(D_i), \tag{22}$$

where $diag(D_i)$ is a vector of the main diagonal elements of D_i.

In order to decrypt the ciphertext, B computes l_i according to (18), t_r according to (5) and (21), E_i according to (19), and

$$(KM_i)^{-1} = (Q \cdot D_i \cdot Q^{-1})^{-1} = Q \cdot D_i^{-1} \cdot Q^{-1}. \tag{23}$$

Then, B retrieves the plaintext:

$$P_i = KM_i^{-1} \cdot (C_i - diag(D_i)). \tag{24}$$

It is appropriate to mention that for computing KM_i we use a diagonal matrix, and only the diagonal entries of D_i are exponentiated to the power l_i, requiring O ($mlog_2 l_i$) multiplications. On the other hand, to get D_i^{-1}, we calculate the inverse of m numbers only. Note also that Q^{-1} and D_i^{-1} are calculated only once. The diagonal elements of D_i^{-1} belong to the group G of numbers co-prime to N. Based on Theorem 10.3 [18] we see that for N=256, $64 = \frac{\varphi(N)}{2}$ is the maximal order of elements of G (odd numbers in Z_{256}). In HCM-EE, we select at least one element in the diagonal with the maximum order to guarantee the maximum period of the diagonal elements. The number of dynamic keys of HCM-EE is estimated as

$$LB \cdot m! \le NDK(HCM\text{-}EE) \le \varphi(N) \cdot m!. \tag{25}$$

where LB is the maximum order of the diagonal elements in D_i. If N is a power of 2, $LB = \frac{\varphi(N)}{2}$.

III. The Proposed Extension of HCM-EE (HCM-EXDKS)

Here, we extend the HCM-EE (HCM-EXDKS) to be more secure and more effective in encryption quality by using dynamic keys together with the dynamic key size for encryption each plaintext block.

The proposed extension assumes the same initial settings and the same encryption-decryption as in HCM-EE. Additionally, we assume the parties share the secret values MAXks and MINksb, which are used to calculate the size for each block by using the pseudo-random numbers generated by (18), MAXks represents the maximum and MINks is the minimum key size. Hence, HCM-EXDKS calculates the key size as follows:

$$KeySize_i = (l_i \bmod s) + MINks, \tag{26}$$

where $s = MAXks - MINks + 1$ and $i \ge 1$. The number of dynamic keys of HCM-EXDKS is estimated as

$$LB \cdot DS \le NDK(HCM\text{-}EEX) \le \varphi(N) \cdot DS, \tag{27}$$

where $DS = sum(used\ Keysize_i!)$.

A. Example

Let N=256, the plaintext to be encrypted is P =”155 140 130 101 207 156 102 235 65 48 32 0 149 76 199 156 255 240 230 201 187 156 142 73 13 17 20 111”, $MAXks = 16$, $MINks$ =5, note that by (18), the set of pseudo-random numbers l contains $l_1 = 157$, $l_2 = 23$, and $l_3 = 121$, the set of eigenvalues E ={75, 171, 91, 215, 99, 95, 111, 91, 115, 95, 81, 7, 47, 89, 105}, the set of a pseudo-random sequence of permutations t contains: $t_1 = (2, 3, 5, 1, 6, 4)$ t_2=(1,2,3,4,5,6,7,8,9,10,11,12,16,15,14,13), and t_3=(2,1,3,4,6,5) .

$$Q = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 12 & 12 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
21 & 11 & 15 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20 & 4 & 19 & 11 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 25 & 19 & 17 & 3 & 10 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
14 & 15 & 17 & 17 & 14 & 15 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
16 & 25 & 0 & 12 & 22 & 13 & 13 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 20 & 22 & 2 & 14 & 3 & 6 & 5 & 13 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 4 & 14 & 19 & 14 & 12 & 10 & 22 & 5 & 20 & 1 & 0 & 0 & 0 & 0 & 0 \\
12 & 8 & 14 & 10 & 10 & 16 & 10 & 10 & 14 & 19 & 2 & 1 & 0 & 0 & 0 & 0 \\
16 & 24 & 10 & 11 & 21 & 23 & 25 & 4 & 0 & 11 & 18 & 22 & 1 & 0 & 0 & 0 \\
24 & 0 & 25 & 3 & 21 & 19 & 1 & 17 & 24 & 2 & 10 & 0 & 14 & 1 & 0 & 0 \\
7 & 16 & 7 & 17 & 11 & 25 & 22 & 24 & 16 & 4 & 9 & 18 & 4 & 25 & 1 & 0 \\
19 & 13 & 15 & 19 & 16 & 25 & 22 & 22 & 20 & 5 & 17 & 3 & 18 & 10 & 1 & 1
\end{bmatrix}$$

$$Q^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
252 & 252 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
41 & 36 & 244 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
172 & 197 & 21 & 253 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
109 & 30 & 243 & 7 & 250 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
73 & 84 & 252 & 178 & 57 & 246 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
215 & 77 & 100 & 154 & 161 & 15 & 253 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
216 & 75 & 15 & 105 & 207 & 57 & 255 & 252 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
48 & 148 & 100 & 144 & 6 & 9 & 22 & 47 & 243 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & 109 & 193 & 70 & 167 & 61 & 133 & 82 & 255 & 236 & 1 & 0 & 0 & 0 & 0 & 0 \\
150 & 252 & 232 & 172 & 192 & 123 & 118 & 13 & 235 & 21 & 254 & 1 & 0 & 0 & 0 & 0 \\
245 & 147 & 242 & 97 & 244 & 104 & 131 & 21 & 111 & 143 & 26 & 234 & 1 & 0 & 0 & 0 \\
48 & 168 & 179 & 201 & 57 & 124 & 194 & 151 & 250 & 244 & 138 & 52 & 242 & 1 & 0 & 0 \\
113 & 146 & 63 & 182 & 21 & 32 & 243 & 141 & 129 & 38 & 57 & 50 & 90 & 231 & 1 & 0 \\
112 & 6 & 160 & 196 & 26 & 18 & 222 & 201 & 106 & 84 & 132 & 79 & 32 & 15 & 255 & 1
\end{bmatrix}$$

Note that by (26), the set of dynamic key sizes (DKS) DKS = {6, 16, 6}.

The first block encryption:

Since $KeySize_1 = 6$, hence, the used matrices will be 6×6, and E_1 will take the first 6 elements from E, therefore $E_1 = \{75, 171, 91, 33, 215, 99\}$, by (19) using the permutation $t_1 = (2, 3, 5, 1, 6, 4)$ and $l_1 = 157$, the new $E_1 = \{155, 139, 247, 187, 115, 161\}$ which represents the diagonal elements of the diagonal matrix D_1, it is worth noting that 6×6 block matrices will be selected from Q and ,Q^{-1} (selection will be from left to right and up to down) which will be used to compute KM_1 according to (20), then the first 6 elements will be selected from the plaintext to represent the first plaintext block P_1 =[155, 140, 130, 101, 207, 156]. By using (22), $C_1 = [116, 143, 85, 194, 208, 25]$. The same steps can be followed to encrypt the second block with length 16 and the third block with length 6. Hence, the ciphertext is "116 143 85 194 208 25 117 68 230 229 163 79 246 19 212 247 204 23 213 188 152 177 165 110 122 214 5 6". The decryption is depicted as follows:

Since the receiver has the matrices Q, Q^{-1}, the set E and the encrypted block number, the receiver can calculate the elements of l according to (18), E_i using (19) to construct D_i^{-1}, the $KeySize_i$ according to (26) and then compute the inverse of the key matrix KM_i by (23).

The first block decryption:

E_1 contains the first 6 elements from E, due to $keySize_1 = 6$, hence, the used matrices will be 6×6, and $E_1 = \{75, 171, 91, 33, 215, 99\}$, by (19) using the permutation $t_1 = (2, 3, 5, 1, 6, 4)$ and $l_1 = 157$, the new $E_1 = \{155, 139, 247, 187, 115, 161\}$, $\bar{E}_1$ contains the multiplicative inverse of the elements in E_1, $\bar{E}_1$ ={147, 35, 199, 115, 187, 79} which represents the diagonal elements of D_1^{-1}. By using 6×6 block matrices selected from Q and $,Q^{-1}$ which will be used to compute KM_1^{-1} according to (23), then the first 6 elements of the ciphertext $C_1 = [116, 143, 85, 194, 208, 25]$ will be decrypted according to (24). Hence, the obtained plaintext P_1 = [155, 140, 130, 101, 207, 156]. The same steps can be followed to get P_2 and P_3 but using key size accordingly.

IV. Simulation Results and Discussion

The simulations are hosted on a Windows XP OS running on a Dell Latitude D630 laptop with Intel(R) Core(TM) 2 Duo 1.8 GHz processor and with 2-GB RAM. The simulation is implemented by Visual studio Environment version 2008. The performance evaluation tool used is C# application, which provides a wide range of profiling instruments for reading and manipulating images. 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=16 when HCM-PT, HCM-H, HCM-HMAC, HCM-EE are used and $m \in [3, ..., 16]$ in the case of HCM-EXDKS.

We examine the encryption quality for three different images containing very large single colour areas: Nike.bmp (Fig. 1), Symbol.bmp (Fig. 2), and Blackbox.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 HCM-PT, HCM-H, HCM-HMAC, HCM-EE, and HCM-EXDKS.

Table 1. ID For Encrypted Images Using HCM-PT, HCM-H, HCM-HMAC, HCM-EE, m=16 And HCM-EXDKS, $m \in [3, ..., 16]$. The smaller ID, the better.

Image/Algorithm	HCM-PT	HCM-H	HCM-HMAC	HCM-EE	HCM-EXDKS
Nike.bmp	23980.79	13171.75	9983.87	2656.62	1676.54
Symbol.bmp	10482.25	5755.68	4830.91	2378.07	1851.66
Blackbox.bmp	34036.28	18511.62	11491.48	3285.25	1742.05
Lena.bmp	10256	10518.66	10469.33	10172.66	10110.66
Girl.bmp	11459.55	10472.61	10336.77	9942.21	9753.75

The quality of encryption of these images is studied by visual inspection (Figs. 1-5) and quantitavely (Table I, used irregular deviation based quality measure ID [9, 20, 21] is explained in the Appendix).

Based on visual inspection, it is obvious that HCM-EXDKS and HCM-EE are better than HCM-PT, HCM-H, and HCM-HMAC in hiding all the features of the image containing large single colour areas (Figs. 1-3).

Based on the numerical evaluation of encryption quality measure ID (Table I, the smaller ID, the better), we note that the proposed extension HCM-EXDKS versus HCM-EE gives better encryption quality. Table I shows also that the proposed extension HCM-EXDKS is more effective in encryption quality than HCM-PT, HCM-H, HCM-HMAC, and HCM-EE. On the other hand, HCM-PT, HCM-H, HCM-HMAC, HCM-EE, and HCM-EXDKS are all good in encrypting images containing many high frequency components (Lena.bmp and Girl.bmp); all the algorithms give nearly the same results.

We examined the encryption time for the Nike.bmp image having $124x124$ pixels and 45KB size. The encryption time measured when applying HCM-PT, HCM-H, HCM-HMAC, HCM-EE, and HCM-EXDKS is shown in Table II. In our implementation, HCM-EE and HCM-EXDKS were used with RC4 [4] for the pseudo-random permutation generator (5), and pseudo-random number generator (18). We implemented HCM-H with SHA-1 [16] since the latter has been used in [10], and the built-in HMAC from C#, HCM-HMAC is used with sha-1. Table II shows that HCM-EXDKS has nearly the same execution time as of HCM-EE but HCM-EXDKS has better encryption quality (Figs. 1-5 and Table I) and roughly is twice better than HCM-H; both HCM-EE and HCM-EXDKS have nearly the same execution time as of HCM-HMAC but HCM-EE and HCM-EXDKS have better encryption quality (Figs. 1-5, and Table I). Table II shows that HCM-NPT is faster than HCM-EE but equations (4) and (24) show that NDK(HCM-EE) is greater than NDK(HCM-PT), hence HCM-EE is more secure than HCM-PT. Inequality (27) shows that NDK(HCM-EXDKS) is greater than NDK(HCM-EE). Hence HCM-EXDKS is more secure and is more effective in the encryption quality than HCM-PT, HCM-H, HCM-HMAC and HCM-EE, and has nearly the same encryption time as HCM-EE and HCM-HMAC

Table 2. Encryption Time (msec) of Nike.bmp With HCM-PT, HCM-H, HCM-HMAC, HCM-EE and HCM-EXDKS.

HCM-NPT	HCM-H	HCM-$HMAC_k$	HCM-EE	HCM-EXDKS
103	*425*	*214*	*200*	*207*

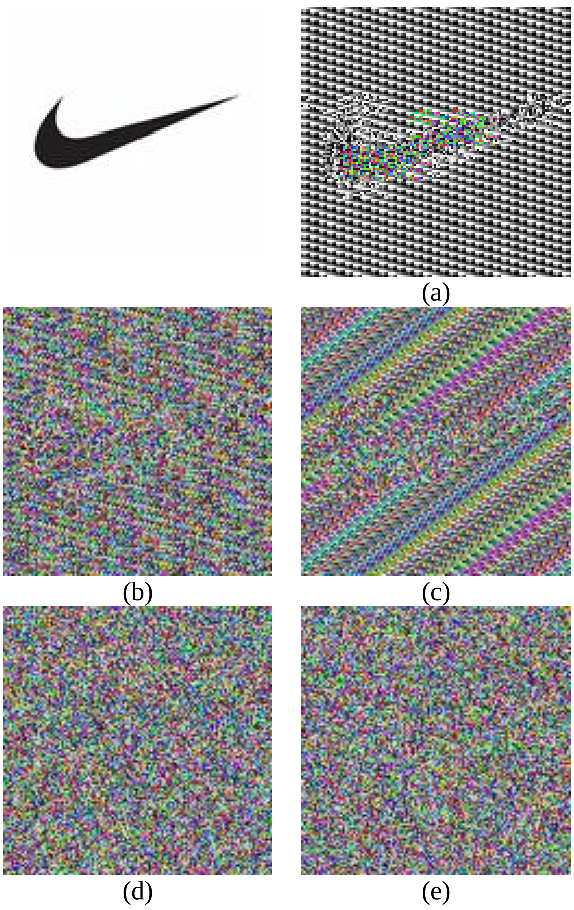


Fig 1:Nike.bmp encrypted by: a) HCM-PT, b) HCM-H, c) HCM-HMAC, d) HCM-EE, e) HCM-EXDKS.

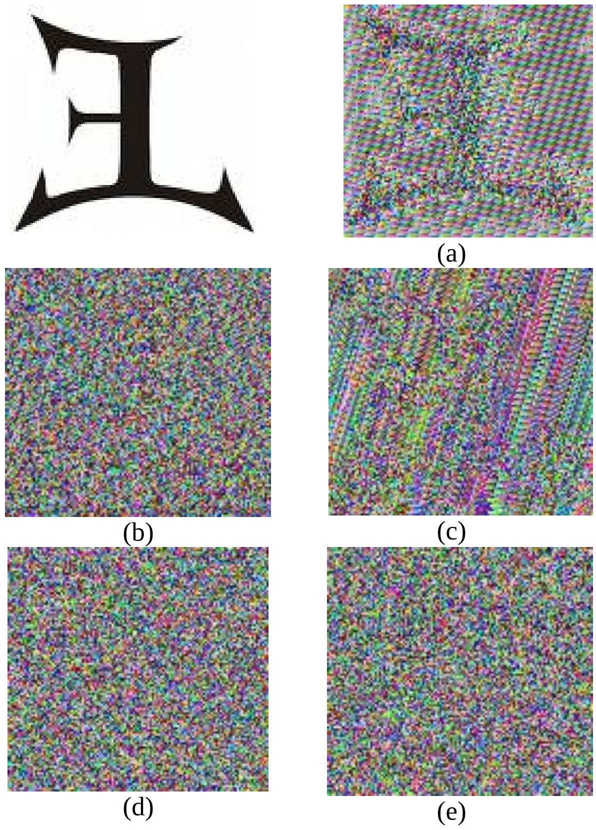


Fig 2: Symbol.bmp encrypted by: a) HCM-PT, b) HCM-H, c) HCM-HMAC, d) HCM-EE, e) HCM-EXDKS.

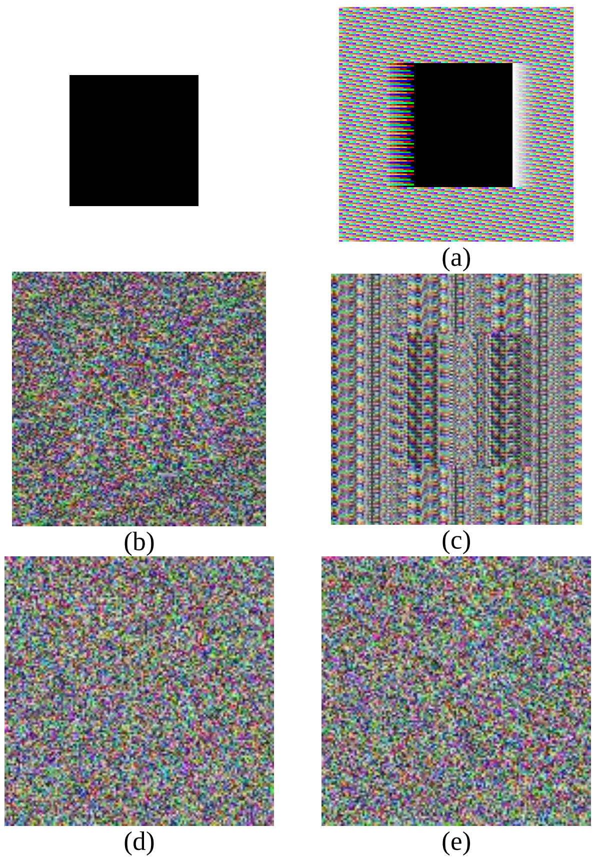


Fig 3: Blackbox.bmp encrypted by: a) HCM-PT, b) HCM-H, c) HCM-HMAC, d) HCM-EE, e) HCM-EXDKS.

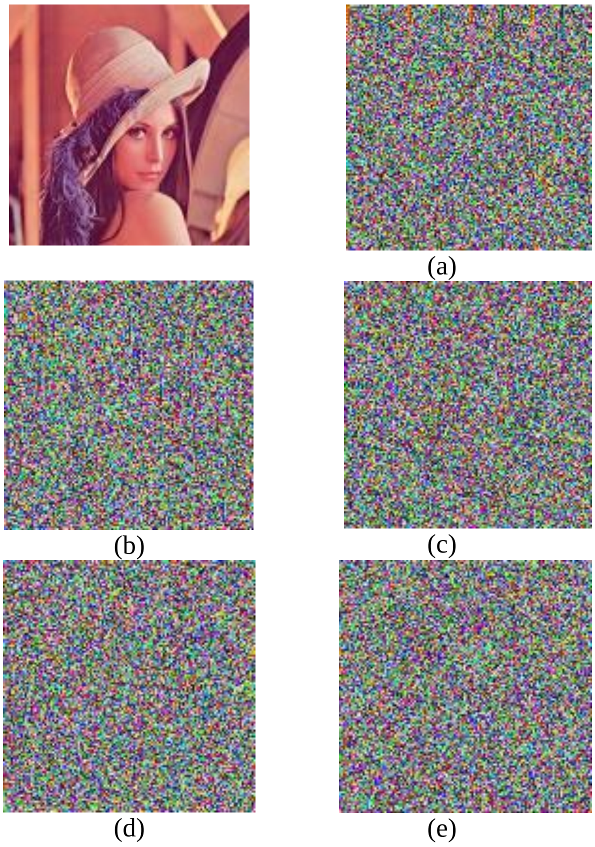


Fig 4: Lena.bmp encrypted by: a) HCM-PT, b) HCM-H, c) HCM-HMAC, d) HCM-EE, e) HCM-EXDKS.

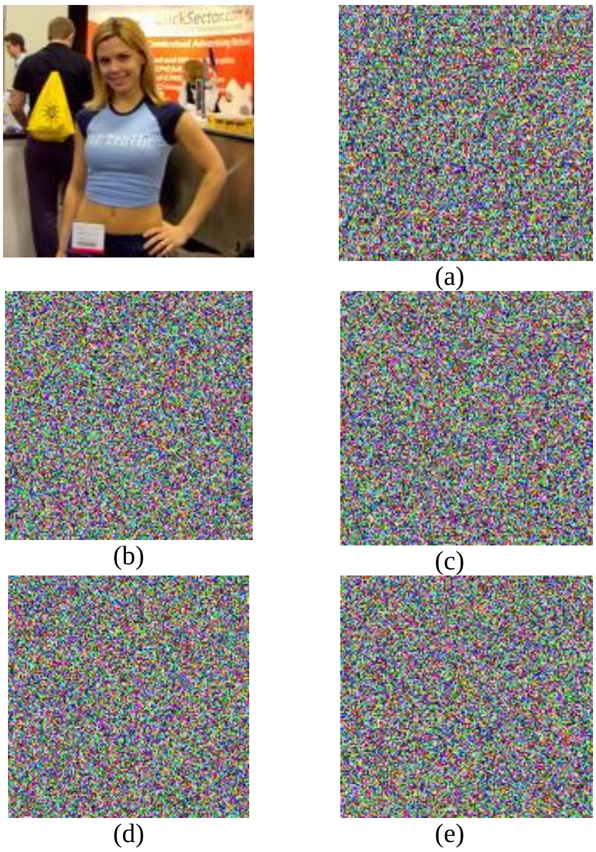


Fig 5: Girl.bmp encrypted by: a) HCM-PT, b) HCM-H, c) HCM-HMAC, d) HCM-EE, e) HCM-EXDKS.

V. Conclusions

Thus far, we have presented an extension of HCM-EE. Versus HCM-EE, the proposed extension HCM-EXDKS resists the known plaintext-ciphertext attack because of the use of dynamically changing key matrices similar to other considered here HC- modifications (HCM-NPT, HCM-EE and HCM-H), but the proposed HCM-EXDKS is the most secure and efficient; HCM-EXDKS uses dynamically changing key together with the key size. HCM-EXDKS is more secure than HCM-EE, HCM-H and HCM-NPT because of the significantly larger number of dynamic keys generated. The proposed HCM-EXDKS is more effective in the encryption quality of RGB images than HCM-EE and HC-known modifications in the case of images with large single colour areas and slightly more effective otherwise.

Appendix

A. Irregular deviation ID quality

Irregular deviation ID quality measuring factor is based on how much the deviation affected by encryption is irregular [9, 20, 21]. This quality measure can be formulated 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:

   $$h = \text{histogram}(D).$$

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:

   $$AC(i) = |hi - DC|.$$

5. Count:

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

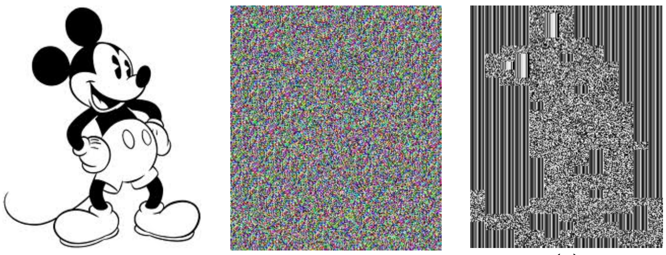
The smaller ID, the better.

B. HCM-EXDKS Versus AES

To give adequate performance comparison, we examine our proposed extension HCM-EXDKS versus other well known algorithms (e.g. AES). We examined the encryption quality of several images. Based on visual inspection, the proposed HCM-EXDKS 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 (Mecy.bmp: Fig. 6, Penguin.bmp: Fig. 7, and bicycle.bmp: Fig. 8). That is, the proposed HCM-EXDKS has advantage in encryption of identical plaintext blocks over the AES.

Table 3: ID for encrypted images using HCM-EXDKS and AES, m=16 and HCM-EXDKS, $m \in [3,...,16]$.

Image/Algorithm	HCM- EXDKS	AES
Mecy.bmp	1872.58	47726.75
bicycle.bmp	7736.79	25031.32
Penguin .bmp	7440.06	20745.34



(a) (b) (c)

Fig 6: a) Mecy.bmp encrypted by: b) HCM-EXDKS, c) AES

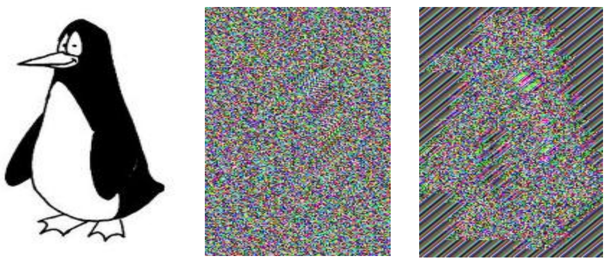


Fig 7: a) Penguin.bmp encrypted by: b) HCM-EXDKS, c) AES.

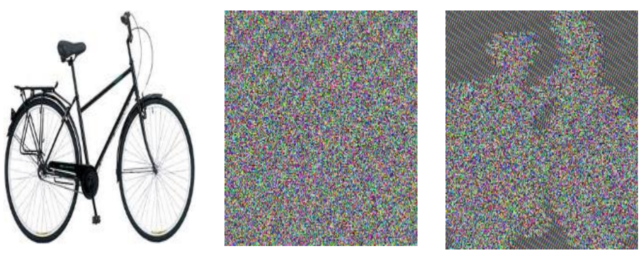


Fig 8: a) Bicycle.bmp encrypted by: b) HCM-EXDKS, c) AES.

REFERENCES

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

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

[3] Overbey, J. Traves, W. and Wojdylo, J. 2005. On the Key Space of the Hill Cipher, Cryptologia, 29(1), 59-72.

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

[5] Saeednia, S. 2000. How to Make the Hill Cipher Secure, Cryptologia, 24(4), 353-360.

[6] Konheim A.G. 2006. Computer Security and Cryptography. John Wiley & Sons: New Jersey, 2007.

[7] Koblitz N. 1987. A course in Number Theory and Cryptography. Springer-Verlag: New York, 64-74.

[8] Chefranov, A. G. 2008. Secure Hill Cipher Modification SHC-M, Proc. of the First International Conference on Security of Information and Networks (SIN2007) 7-10 May 2007, Gazimagusa (TRNC) North Cyprus, Elçi, A., Ors, B., and Preneel, B. (Eds.) Trafford Publishing, Canada. 34-37.

[9] Ismail A.I., Amin M, Diab H. 2006. How to Repair the Hill Cipher. J. Zhejiang Univ Sci. A; 7(12): 2022-2030

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

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

[12] Romero Y.R, Garcia R.V, et al. 2007. Comments on How to Repair the Hill Cipher. Journal of Zhejiang University Science A ; 9(2): 211-214.

[13] Li CD, Zhang D, Chen G. 2008. Cryptanalysis of an Image Encryption Scheme Based on the Hill Cipher. Journal of Zhejiang University Science A; 9: 1118-1123.

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

[15] Rivest R, The MD5 Message-Digest Algorithm. Internet RFC 1321, April 1992.

[16] Federal Information Processing Standard (FIPS) 180-2, 2002, Secure Hash Standard, NIST, U. S. Department of Commerce.

[17] Schneier B., 1996. Applied cryptography: Protocols, Algorithms, and Source Code in C (2nd edn), John Wiley & Sons: New York.

[18] Apostol T.M. 1976. Introduction to Analytic Number Theory Springer.

[19] Galvin, W. P. 1984. Matrices with Custom-Built Eigenspaces, this MONTHLY, 91, 308-309.

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

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

Ahmed Y. Mahmoud received a BSc. in Computer Science from Al-Zaytoonah University, Amman, Jordan in 1997, an MSc. in Applied Mathematics and Computer Science and PhD in Computer Engineering from Eastern Mediterranean University, North Cyprus, in 2001 and 2012, respectively. From 2001 to 2006 he was a Lecturer in the Computer Science Department at Al-Azhar University, Gaza Strip, Palestine. From September 2006 to September 2011 he was with the Computer Engineering Department at Eastern Mediterranean University, Famagusta, North Cyprus. Since January 2012 he has been an Assistant professor in the Information Technology Department at Al-Azhar University, Gaza Strip, Palestine. His research interests are in the areas of information security, discrete geometry, parallel programming, and distributed systems.

Alexander G. Chefranov received a diploma of Engineer in Applied Mathematics, PhD, and DSc from Taganrog State Radio-Engineering University, Russia, in 1978, 1984, and 1998, respectively. Since 1999 he has been a Professor of Software Engineering Department of the Institute of Technology, South Federal University, Taganrog, Russia, and, since 2002, he has been an Associate Professor of the Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus. His research interests are in the areas of information security, parallel processing, real-time systems, database management systems, and scientific computing.

How to cite this paper: Ahmed Y. Mahmoud, Alexander G. Chefranov,"A Hill Cipher Modification Based on Eigenvalues Extension with Dynamic Key Size HCM-EXDKS", IJCNIS, vol.6, no.5, pp.57-65, 2014. DOI: 10.5815/ijcnis.2014.05.08