Hill Cipher Modification Based on Eigenvalues HCM-EE

Ahmed Y. Mahmoud Alexander G. Chefranov
Eastern Mediterranean University, North Cyprus,
Computer Engineering Department
Ahmed.mahmoud@emu.edu.tr Alexander.chefranov@emu.edu.tr

ABSTRACT
We propose and analyze a new modification of the Hill cipher, HCM-EE, generating dynamic encryption key matrix by exponentiation that is made efficiently with the help of eigenvalues. Security of HCM-EE is provided by the use of a large number of dynamic keys. Experimental results show that HCM-EE is from two to seven times more effective in the encryption quality of RGB bitmap images than Hill cipher and its known modifications in the case of images with large single colour areas, and slightly more effective otherwise. It is two times faster than HCM-H, having the best encryption quality among Hill cipher modifications compared versus HCM-EE.

Categories and Subject Descriptors
D.4.6, [Security and Protection]: - *cryptographic controls*. K.6.5 [Security and Protection]: - *cryptographic controls*. E.3 [Data Encryption]: - standards. H.2.0 [General]: - *security, integrity, protection*.

General Terms: Security, Algorithms, Performance

Keywords
Hill cipher, matrix, eiqenvalue, exponentiation, pseudorandom number, dynamic key, image encryption

1. INTRODUCTION
The Hill cipher (HC) [5, 6] is a well known symmetric cryptosystem using a simple linear transformation. It is very attractive due to its simplicity and high throughput [8, 9], but it can be broken by the known plaintext-ciphertext attack (KPCA) [10]. It has a large key space calculated in [1, 8] for various dimensions. HC modification [9], HCM-PT, uses a dynamic key matrix obtained by random permutations of rows and columns from the master key matrix and transfers an HC-encrypted permutation to the receiving side. Thus, in HCM-PT, each plaintext vector is encrypted by a new key matrix that prevents the KPCA on the vectors. But permutations in HCM-PT are transferred HC-encrypted, which means that master key matrix can be revealed by the KPCA on the permutations [7]. Proposed in [2] HCM-PT modification, HCM-NPT, works as HCM-PT does, but does not transfer permutations; instead, both communicating parties use pseudo-random permutation generator, and only the number of the necessary permutation is transferred to the receiver. It has good computational complexity but the number of its dynamic keys is the same as for HCM-PT.

In [7], another HC modification, HCM-H, is proposed. It uses a one way hash function applied to an element of the master key matrix picked up randomly by the sender to get a key matrix, and a vector added to the product of the key matrix and a plaintext. The use of hash functions in [7] makes it rather computationally hard.

In this paper, we propose the Hill cipher modification, HCM-EE, based on the use of pseudo-random eigenvalues to construct a key matrix [4] and modify it for each new plaintext. It has two times better computational complexity than that of HCM-H [7] and is more secure than HCM-H and HCM-NPT [2] because of the significantly larger number of dynamic keys. The HCM-EE is six to seven times better in encryption quality than HC, HCM-PT, and HCM-NPT, and two times better than HCM-H in the case of images with large single colour areas. In the other considered cases, the quality of the compared ciphers is practically the same, but HCM-EE has slightly better quality.

The paper is organized as follows. Section 2 briefly discusses HC, HCM-PT, HCM-NPT, and HCM-H, and number of dynamic keys used is evaluated for HC modifications. Section 3 presents HCM-EE, and the number of dynamic keys for it is estimated. In Section 4, comparison of the proposed scheme, HCM-EE, versus HC modifications is given; estimates of their execution time and experimental results on image encryption are presented. Conclusion is in the Section 5.

2. 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 to be made *mod N*, where *m* (block size) and *N* (alphabet cardinality) are selected positive integers (e.g., *N=256* for gray scale image). Also, we assume in the paper 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, and HCM-NPT, and then we describe HCM-H. When HC is used, *A* and *B* share a non-singular invertible key matrix $K_{m\times m}$. To encrypt an *m*-component plaintext vector, *P*, into the ciphertext vector, *C*, *A* performs

$$C = K \cdot P \,. \quad (1)$$

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

$$P = K^{-1} \cdot C \,, \quad (2)$$

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

$$gcd(det(K) \bmod N, N) = 1 \,. \quad (3)$$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIN'09, October 6–10, 2009, North Cyprus, Turkey.
Copyright 2009 ACM 978-1-60558-412-6/09/10...$10.00.

HCM-PT [9] differs from (1), (2) in the following. To encrypt a plaintext P, A selects a permutation, t, randomly over Z_m, and performs the permutation over the rows and columns of a key matrix K which produces a new key-matrix $K_t=t(K)$. HCM-PT encryption is then performed by (1), but using K_t 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 $(K^{-1})_t = (K_t)^{-1}$ [9] 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 [2] 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 pseudorandom 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 [7], another HC modification, HCM-H, works as follows. The sender, *A,* and the receiver, *B,* share an invertible matrix K_{mxm}. To encrypt the plaintext $P = [p_1, p_2, ..., p_m]$, 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)

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

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

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

3. THE PROPOSED HCM-EE SCHEME

In HCM-EE, A selects a set $E = \{e_1, e_2, ..., e_m\} \subset Z_N - \{0\}$ of eigenvalues of the matrix to be constructed, e_j is relatively prime to N, $1 \le j \le m$, and constructs an invertible matrix Q. Then, the sender A calculates the key matrix K [4]:

$$K = Q \cdot D \cdot Q^{-1}, \tag{11}$$

where D is a diagonal matrix, diagonal elements of which are the 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 pseudorandom numbers $l = \{l_1, l_2, ..., l_n\} - \{0\}$ by (12), and *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

$$l_i = PRNG(SEEDl, i) \neq 0, \tag{12}$$

where $1 \le i \le n$, n is the number of blocks. Then, using (13), A calculates

$$KM_i = Q \cdot \left[t_r(D)\right]^{l_i} \cdot Q^{-1}, \tag{13}$$

where $i = \varphi(N) \cdot r + j$, and $\varphi(N)$ is the Euler function [10]. The random permutation, t_r, is obtained by (5). The plaintext P_i is encrypted by (14)

$$C_i = KM_i \cdot P_i + diag(t_r(D)), \tag{14}$$

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

In order to decrypt the ciphertext, B computes l_i according to (12), calculates t_r according to (5), and finds $(KM_i)^{-1}$ from (15) as

$$(KM_i)^{-1} = (Q \cdot \left[t_r(D)\right]^{l_i} \cdot Q^{-1})^{-1} = Q \cdot \left[t_r(D^{-1})\right]^{l_i} \cdot Q^{-1}. \tag{15}$$

Then, B retrieves the plaintext by (16)

$$P_i = KM_i^{-1} \cdot (C_i - diag(t_r(D))). \tag{16}$$

It is appropriate to mention that for computing KM_i we use a diagonal matrix, and only the diagonal entries of D shall be exponentiated to the power l_i, requiring $O(m log_2 l_i)$ multiplications. On the other hand, to get D^{-1}, we calculate the inverse of m numbers only. Note also that Q^{-1} and D^{-1} are calculated only once. The number of dynamic keys of HCM-EE is

$$NDK(HCM\text{-}EE) = \varphi(N) \cdot m!. \tag{17}$$

4. EXPERIMENTS ON THE CIPHERS COMPARISON

We developed programs for simulating the encryption algorithms in Matlab 7.0 on an Intel(R) Core(TM) 2 Duo 1.8 GHz processor with 2-GB RAM and Windows XP.

In our experiments, several RGB images are evaluated. Three different images containing very large single colour areas are used: Nike.bmp (Fig. 1), Symbol.bmp (Fig. 2), and Blackbox.bmp (Fig. 3). Also we examined the HC, HCM-PT, HCM-H, and HCM-EE on an image containing many high frequency components, Lena.bmp (Fig. 4). We use encryption quality measure EQM [11], $1 \ge \delta(f, f_1) \ge 0$, where f and f_1 are an original and encrypted images respectively

with (x,y) pixels; the greater EQM is, the better the quality of encryption. The $\delta(f, f_1)$ *is* calculated by (18)

$$\delta(f, f_1) = \frac{\sum_{i=1}^{x}\sum_{j=1}^{y} S_{ij}}{255^2 \times x \times y}$$

$$S_{ij} = K_1(i,j) + K_2(i,j) + K_3(i,j) + K_4(i,j)$$

$$K_1(i,j) = \left|[f_1(i-1,j) - f_1(i,j)]^2 - [f(i-1,j) - f(i,j)]^2\right| \quad (18)$$

$$K_2(i,j) = \left|[f_1(i+1,j) - f_1(i,j)]^2 - [f(i+1,j) - f(i,j)]^2\right|,$$

$$K_3(i,j) = \left|[f_1(i,j-1) - f_1(i,j)]^2 - [f(i,j-1) - f(i,j)]^2\right|$$

$$K_4(i,j) = \left|[f_1(i,j+1) - f_1(i,j)]^2 - [f(i,j+1) - f(i,j)]^2\right|$$

An RGB image is stored as a three-dimensional data array, *rgbimg[x,y,3]* (three layers), each layer representing one colour. We reshape *rgbimg[x,y,3]* into the one-dimensional array *onedim[3·x·y]* and use the block size *m*=16. The quality of encryption of these images is studied by visual inspection (Figs. 1-4) and quantitavely (Table 1). Based on visual inspection and EQM, we note that HCM-EE is six to seven times better than HC, HCM-PT, and about two times better than HCM-H in hiding features of all the images containing large single colour areas (Table 1, rows 1-3). On the other hand, HC, HCM-PT, HCM-H and HCM-EE are all good in encrypting images containing many high frequency components; all the algorithms give nearly the same results but the proposed HCM-EE is the most effective (Table 1, rows 4-5).

Table 1. EQM for encrypted images using HC, HCM-PT HCM-H, and HCM-EE, *m=16*

Image/Algorithm	HC	HC-PT	HCM-H	HCM-EE
Nike.bmp	*0.0917*	*0.10284*	*0.3548*	*0.6779*
Symbol.bmp	*0.1123*	*0.1314*	*0.3817*	*0.7653*
Blackbox.bmp	*0.1121*	*0.1143*	*0.4111*	*0.7721*
Lena.bmp	*0.6530*	*0.6541*	*0.6565*	*0.6673*
Girl.bmp	*0.6539*	*0.6547*	*0.6553*	*0.6601*

Encryption time was measured when applying the HCM-EE and HCM-H to the Symbol.bmp image. In our implementation, HCM-EE was used with RC4 [10] as a pseudo-random number and permutation generator in (5), (12). We implemented HCM-H with SHA-1[3] since the latter has been used in [7]. Symbol.bmp has 118x109 pixels and is 37.9 KB-sized. HCM-EE was found to be roughly two times faster than HCM-H (Table 2).

The key space of HCM-EE and HCM-H is the same as that of HC, while *NDK(HCM-EE)* (17) is greater than *NDK(HCM-H)* (10). Hence HCM-EE is more secure than HCM-H.

Table 2. Time (sec) of encryption of Symbol.bmp with HCM-EE and HCM-H.

Block Size, *m*	HCM-EE	HCM-H
3	*2.718*	*5.539*
16	*0.3143*	*0.7247*

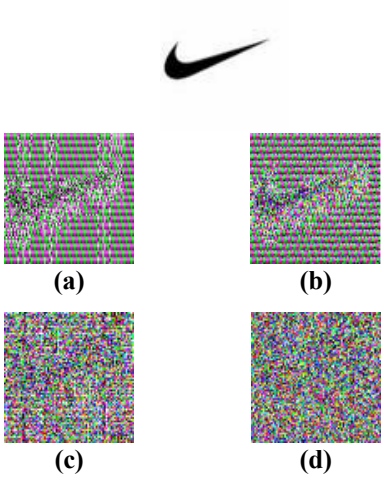

Figure 1. Nike.bmp encrypted by: a) HC, b) HCM-PT, c) HCM-H, d) HCM-EE.

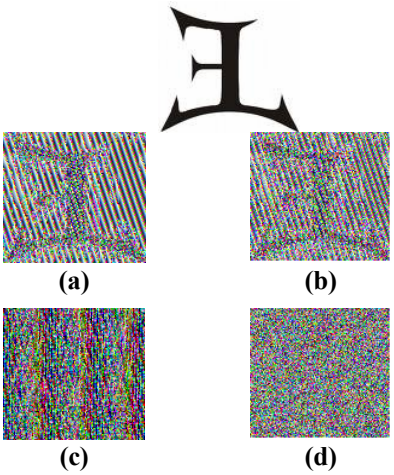

Figure 2. Symbol.bmp encrypted by: a) HC, b) HCM-PT, c) HCM-H, d) HCM-EE.

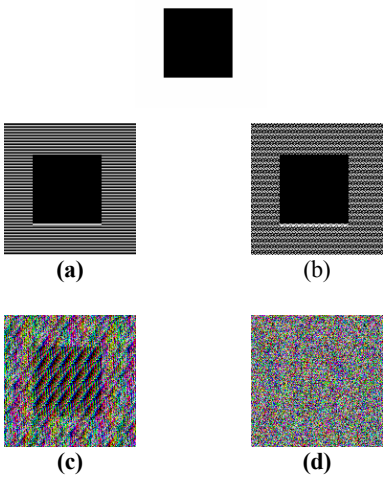

Figure 3. Blackbox.bmp encrypted by: a) HC, b) HCM-PT, c) HCM-H, d) HCM-EE

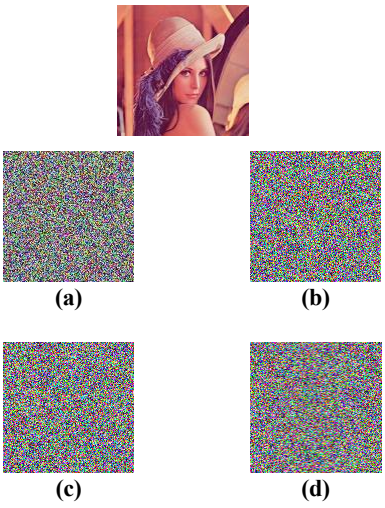

Figure 4. Lena.bmp encrypted by: a) HC, b) HCM-PT, c) HCM-H, d) HCM-EE

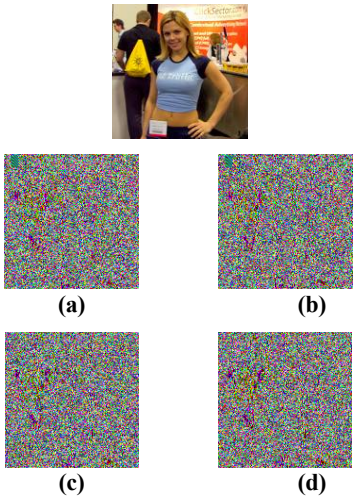

Figure 5. Girl.bmp encrypted by: a) HC, b) HCM-PT, c) HCM-H, d) HCM-EE

5. CONCLUSION

Thus far, we have presented a new HC modification, HCM-EE, based on the use of eigenvalues for matrix exponentiation to a pseudo-random power for generating a new key matrix for each plaintext block. It resists the known plaintext-ciphertext attack because of the use of dynamically changing key matrices similar to other considered here HC modifications (HCM-PT, HCM-NPT, and HCM-H), but the proposed scheme is the most secure and efficient. HCM-EE is more secure than HCM-H and HCM-NPT because of the significantly larger number of dynamic keys generated ((17) versus (10) and (4)). HCM-EE is two to seven times more effective in encryption quality than HC, HCM-PT, and HCM-H (Table 1) for images with large single colour regions, and has slightly better quality otherwise. HCM-EE is two times faster than HCM-H (Table 2), the latter having the best EQM among HC, HCM-PT, and HCM-H (Table 1).

6. ACKNOWLEDGMENTS

The authors thank unknown reviewers for their valuable comments.

7. REFERENCES

[1] Bauer, C. and Millward, K. 2007. Cracking Matrix Encryption Row by Row, Cryptologia, 31(1), 76-83.

[2] 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.

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

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

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

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

[7] 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.

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

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

[10] Stallings, W., 2006, Cryptography and Network Security, 4th Edition. Prentice Hall, Upper Saddle River.

[11] Yu, X. Y. Zhang, J. et al. 2006. A New Measurement Method of Image Encryption. Journal of Physics, 48(1), 408-411.