International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I11P201

Volume 70 Issue 11, 1-9, November 2022

Original Article

A Novel Hash Functions for Data Integrity Based on Affine Hill Cipher and Tensor Product

Ahmed Y. Mahmoud

Department of Information Technology, Faculty of Engineering and Information Technology, Al Azhar University-Gaza, Palestine

Corresponding Author : ahmed@alazhar.edu.ps

Received: 24 July 2022 Revised: 05 September 2022 Accepted: 22 September 2022 Published: 26 November 2022

Abstract *- Nowadays, Cryptographic hash functions as a part of cryptosystems play an essential role in information security. It is aimed at providing confidentiality, authentication, integrity and non-repudiation. Thus the importance of hash functions and their use in several applications showed the necessity of strong and efficient hash functions. The hash function only works in one direction and cannot be reversed. In this paper, we present two new efficient and secure hash functions; the first hash function is based on affine Hill cipher transformation; it uses non-invertible matrix multiplication; the second hash is based on the tensor (Kronecker) product. The proposed schemes depend on matrix multiplication and addition over Z_N; essentially, they use a non-invertible matrix and utilize the properties of affine ciphers. The analysis of the proposed hash functions proves that the proposed schemes satisfy the requirements of hash functions.*

Keywords *- Hash function, Data integrity, Affine cipher, Matrix cipher, Tensor product, Kronecker product.*

1. Introduction

One of the most well-known and significant methods in the realm of cryptography is the hash function. The contemporary cryptographic hash functions take an arbitrary length input and produce a fixed length "unique pattern/fingerprint". At the dawn of the development of cryptographic hash functions, the leading approach was algorithmic. Accordingly, the main concern was to create algorithms which are a combination of mathematics and computer science. Cryptographic hash functions have added many security characteristics, particularly beneficial and valuable for computer science and engineering applications, among other disciplines.

Moreover, the hash function uses one-way transformation, which helps avoid revealing the hashed value; it is very useful for verifying data integrity and authentication. In recent years, hash functions have been considered with extreme concentration and developed by many researchers (see, for instance, [1] and the bibliography therein). The main goal is to establish and create a simple (straightforward), efficient and robust algorithm.

On the other hand, many researchers have devoted their effort to developing new algorithms and fixing the drawbacks in the existing ones. Hill cipher (HC), invented by Lister Hill [2, 3], is well known in cryptography. The HC is considered the inventor of symmetric encryption algorithms; it is strong against brute-force and statistical attacks.

Nonetheless, it is susceptible to a known plaintext-ciphertext attack (KPCA). Several researchers have proposed improvements to the Hill cipher to address this weakness and make it more secure [4]–[9]. The main operation of Hill-cipher is matrix manipulation; it multiplies a plaintext vector by a key matrix to get the ciphertext.

On the other hand, it multiplies the ciphertext vector by the inverse of the key matrix to get the plaintext. It is attractive due to its simplicity and high throughput [10]–[12]. The use of HC requires the existence of K^{-1}; note that K^{-1} sometimes does not exist. The non-invertible matrices are not eligible as key matrices in the HC algorithm [12].

In 1990, a tensor-theoretic enhancement to improve the security of the basic Hill system was proposed [9], where a tensor (Kronecker) product is used to increase the block size. The scheme uses *m x m* (m by m) invertible matrix overZ_N; it is expanded to an invertible matrix of order m^3 by using tensor products.

Abu Taha et al. [1] take the opportunity to benefit from the HC simplicity and the non-invertible matrix. They proposed a one-way Hash algorithm primarily based on the non-invertible matrix. Although their design is ineffective, matrix multiplication is used in the Abu Taha et al. hash algorithm. Abu Taha et al. Scheme has numerous security flaws due to the following. In fact, it is inefficient in hashing

the exact/identical plaintext blocks because the same key is used for hashing all plaintext blocks, which makes it vulnerable to statistical analysis.

Furthermore, if all characters in a plaintext block are zero, then the plaintext block $P_i = C_i$ ciphertext block is due to matrix multiplication. The zero plaintext blocks are transformed to zero. Based on the former reasons, it is clear that the proposed one-way hash function may become problematic for identical plaintext blocks, zero plaintext, and grayscale images containing a significant portion of black pixels or large areas of a single color.

This paper's primary goal is to propose and introduce a novel hash algorithm variant of Abu Taha et al.'s scheme [1], which overcomes all of its security flaws. Our proposed Hash functions use the affine Hill cipher and the tensor product. The main idea of an affine cipher is to use multiplication and addition modulo N, where N is a positive integer, to create a more complicated substitution [25][26]. The proposed hash functions use a non-invertible matrix for multiplication and use the output of the previous round or select a column or row vector randomly for addition over modulo purposes. The selection of row/column is done based on a pseudo-random number generator (PRNG); a pseudo-random permutation (PRPerm) is generated for hashing different blocks.

The paper is divided into the following sections. Section 2 contains basic concepts, definitions of encryption, decryption, symmetric, asymmetric encryption algorithms, hash function and requirements of the hash function. Section 3 overviews the proposed hash function based on affine hill cipher HFB-AHC and a practical example is introduced in Section 4. The proposed hash function based on the tensor product HFB-TP is introduced in Section 5. In Section 6, the proof of hash function requirements is presented and discussed. Section 7 discusses the security analysis. Section 8 has the conclusion.

2. Encryption Algorithm vs Hash Algorithm

There are two classes of encryption algorithms, asymmetric encryption (a terminology for public key encryption) uses two separate keys, public and private [15] (e.g. RSA and ElGamal). Symmetric encryption algorithm (e.g. AES, DES, and Blowfish) which uses the same key securely exchanged between the sender and the receiver [16]–[18]. Usually, symmetric encryption is used for large sizes of data such as images and text files. Asymmetric encryption is used for small data, such as encryption keys. According to [10], encryption is defined as transforming/converting the original message/information/ data into an unreadable message; the input of the encryption algorithm is called plaintext, and the produced output is referred to as ciphertext, the general form of the encryption process is illustrated by (1)

$$C_i = E_k(P_i) \tag{1}$$

Where C denotes the ciphertext, E represents the encryption algorithm, K is the encryption key, P is the original message, and $1 \leq i \leq n$, n is the number of blocks.

The plaintext is defined/known as the original message. Decryption is defined as the reverse of the encryption operation; it reveals the plaintext from the ciphertext, the general form of the decryption process is depicted by (2) [10]

$$P_i = D_k(C_i) \tag{2}$$

Where C denotes the ciphertext, D is the Decryption algorithm, K is the decryption key, P_i is the block of the original message, and $1 \leq i \leq n$, n is the number of blocks.

On the other hand, the hash algorithm is employed for one-way encryption, i.e., the original message or plaintext cannot be revealed back from the hashed value [19]. The hash function output is used for data integrity, digital signatures, and authentication [20][21]. It is appropriate to mention that the one-way hash function is a mechanism that transforms a variable string length into a fixed length and the output length is shorter than the input length (e.g., SHA256, SHA512, MD4, and MD5) [10]. To have an effective hash algorithm, the following four requirements must be satisfied: (1) applicable to any arbitrary size of data, (2) produce a fixed size of data, (3) simple to compute for any arbitrary data, and (4) one-way property [10].

3. Overview of Affine Cipher, Hill Cipher, Tensor Product and Affine Hill Cipher

The basic idea of AHC depends mainly on the combination of Affine and Hill cipher, respectively. For deep understanding, we first recall AC and HC, Tensor Product, and then present the AHC.

3.1. Affine Cipher AC

The AC is one of the well-known substitution ciphers. The main operation of AC is multiplication and addition. When AC is used, the two parties of the communication, (sender A) and (receiver B), share a secret multiplicative key and an additive key b. The sender A transforms plaintext to ciphertext by applying (3)

$$c = a \,.\, p + b \bmod N \tag{3}$$

The receiver, B, decrypts the ciphertext by applying (4)

$$p = a^{-1} \,.\, c + b \bmod N \tag{4}$$

Where $a, b \in Z_N$, N is (alphabet cardinality) and a^{-1} is the multiplicative inverse of a over Z_N, for the existence of a^{-1}, a must be relatively prime (co-prime) to N, i.e., $gcd(a, N)=1$,

gcd stands for the greatest common divisor. Note that when *a*=1, the Affine cipher works similar to the Caesar cipher [10]. Affine cipher is not secure; it is susceptible and vulnerable to the frequency analysis attack and has the same essential drawbacks as the substitution ciphers.

3.2. Hill Cipher HC

The fundamental concept of the HC is to break (divide) the plaintext characters into blocks of length *m.* HC assumes the dimension (size) of the secret key matrix is *m x m* and then transforms every block of plaintext characters into a vector of integers by the selected alphabet. The transformed plaintext block is multiplied by *m x m* key matrix. The ciphertext message is then created when the obtained results are transformed into letters.

When HC is used, *m x m* square invertible matrix K is exchanged securely between the sender *A*, and a receiver *B*; *K* must be invertible. For the existence of K^{-1}, K must satisfy the following: det(K) ≠ 0 and det(K) must be co-prime to *N*; more precisely, K must satisfy (5)

$$\begin{matrix} det(K)\neq 0, \text{ and} \\ gcd(det(K) \bmod N, N)=1 \end{matrix} \quad (5)$$

Where *m* represents the block size and *N* (alphabet cardinality), *det(k)* stands for the determinant of *K*, and *gcd* is the greatest common divisor. The encryption is achieved by applying (6)

$$C = K \times P \bmod N \quad (6)$$

The decryption is achieved by applying (7)

$$P = K^{-1} \times C \bmod N \quad (7)$$

Where P represents plaintext, C denotes the ciphertext, K is the key matrix, K^{-1} stands for the inverse of K, and *N* is the alphabet cardinality.

It is worth mentioning that diffusion is a feature of the HC; changing a single letter in the plaintext will affect many letters throughout the ciphertext. Frequency testing becomes more challenging and difficult to implement with the presence of diffusion features. On the other hand, confusion is another characteristic that HC possesses. Each letter in the ciphertext depends on many key components (elements). Consequently, it is impossible to calculate the key part by part. HC is secure against ciphertext-only attacks but suffers when plaintext-ciphertext attacks are applied.

3.3. Tensor Product TP

The scheme proposed in [9] is described below. Given an arbitrary plaintext message *P* of length *L*, defined over an alphabet of order *N* and a non-singular matrix K_{mxm} in Z_N, proceed as follows:

1. Convert P as in the Hill system, each character in the alphabet is assigned a unique integer in {0, 1,…, N-1}, (e.g., N=26 for the English alphabet, and N=256 for grayscale images).

2. Divide the plaintext into b blocks of length m3 and load the plaintext $P_{l_{m\times m\times m}}$ into a rank-three tensor by some predetermined method,
   where b = ceil(L/m^3) and $1 \le l \le b$. It is noticed that if the length L is not a multiple of m^3, the last plaintext block must be padded with bm^3 - L extra characters.

3. Create ciphertext cube C_l, entry-by-entry, by using (8)

$$[C_l]_{ij}^{u} = \sum_{r=1}^{m}\sum_{s=1}^{m}\sum_{t=1}^{m} k_{ri}\cdot k_{sj}\cdot \overline{k_{ut}} \cdot [P_l]_{rs}^{t} \quad (8)$$

   where $K = (k_{ij}), K^{-1} = (\overline{k_{ij}})$,
   $(k_{ri}\cdot k_{sj}\cdot \overline{k_{ut}}) = K \otimes K^{T} \otimes K^{-1}$
   and $1 \le i \le m$, $1 \le j \le m$, $1 \le u \le m$, $1 \le r \le m$, $1 \le s \le m$, $1 \le t \le m$.

4. Reassemble C_l in the Hill system.

3.3.1. Attacks on the Tensor Product Scheme

It is mentioned in [9] that equation (8) can be rewritten as

$$\begin{pmatrix} c_{11}^{1} \\ \vdots \\ c_{mm}^{m} \end{pmatrix} = \begin{pmatrix} k_{11}k_{11}\overline{k_{11}} & \cdots & k_{m1}k_{m1}\overline{k_{1m}} \\ \vdots & \ddots & \vdots \\ k_{m1}k_{1m}\overline{k_{m1}} & \cdots & k_{mm}k_{mm}\overline{k_{mm}} \end{pmatrix} . \begin{pmatrix} p_{11}^{1} \\ \vdots \\ p_{mm}^{m} \end{pmatrix} \quad (9)$$

Furthermore, the most natural attack against the scheme is a known plaintext-ciphertext attack. From equations (8) and (9), the proposed scheme succumbs to a known plaintext-ciphertext attack as in the original Hill cipher. In [9], section 1-c, it is recognized that the Hill cipher is vulnerable to plaintext-ciphertext attack; in [9] - section 4, it is claimed that the proposed there encryption scheme based on the use of the tensor (Kronecker) product improves the security. From equation (9), one can easily notice that the proposed system operates as the original Hill cipher. Regardless of how the matrix key encryption is constructed and the plaintext representation, the proposed scheme is still vulnerable to the known plaintext-ciphertext attack. As it has been recognized and described in [9] that (8) can be viewed as equation (9), and equation (9) can be rewritten as depicted in equation (10)

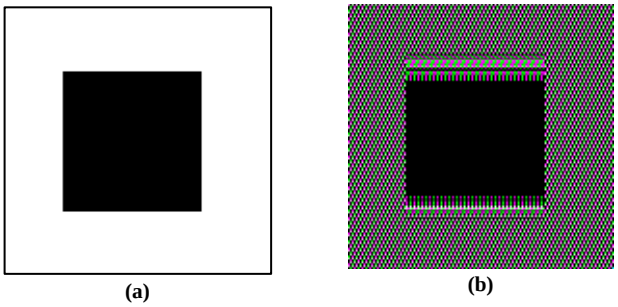
$$C = K_e \cdot P \quad (10)$$

Which is exactly the same as the traditional Hill cipher; if the encryption is performed by using the same key for several messages, then an opponent needs to capture m^3 pairs of column vectors of plaintext and ciphertext to be able

to determine$K_e = K \otimes K^T \otimes K^{-1}$; for example, using an 8 × 8 key matrix, as suggested in [9], it would be able to encrypt blocks of 512 characters. Moreover, implementing a plaintext ciphertext attack is feasible if the opponent collects 512 blocks.

Despite the key matrix size expansion up to m^3as stated in [9], the proposed enhancement [9] is still susceptible and vulnerable to the known plaintext-ciphertext attack. On the other hand, it fails to encrypt a plaintext block of the fixed value zero, as illustrated below (see Figure 1).

A second attack can be implemented if the scheme is implemented for encrypting grayscale images having large portions of pixels in black color or sparse messages, i.e., if all the pixels/ characters in a plaintext block are zeros, then $P_i=C_i=0$. Since black pixels are often mapped to zero in grayscale, the suggested approach may present issues for images with a significant amount of black region. This situation is illustrated in Figure 1.



(a) **(b)**

Fig. 1 Encryption an image black background image. (a) Original image, (b) Encrypted image by Tensor Product Scheme

3.4. Affine Hill Cipher AHC

AHC combines both the affine cipher and the Hill cipher [10]. AHC differs from the original affine cipher in the following. To encipher a plaintext *P*, AHC uses the same initialization as in HC; it also uses a vector of m elements, where m is the block size. The encryption is achieved by applying (11)

$$C = K \times P + V \, mod \, N \tag{11}$$

and the decryption is achieved by applying (12)

$$P = K^{-1} \times C + V \, mod \, N \tag{12}$$

Where *P* represents plaintext, *C* represents the ciphertext, *K* is the encryption key matrix, *K* must satisfy (5), K^{-1} is the inverse of *K* over the modulo *N*, *V* is a vector of *m* elements, and *N* is the alphabet cardinality.

3.5. Abu Taha et al Scheme

Abu Taha e.t. al [1] work similar to the HC and use the same parameters, except it is different from the used key. HC uses an invertible matrix, whereas Abu Taha e.t. al scheme used a non-invertible matrix. The scheme is depicted as follows:

$$H(P) = K \times P \, mod \, N \tag{13}$$

Where *H* denotes the hash function, *H(P)* is the computed hash value, *P* is the plaintext block, K is a non-invertible matrix, and *N* is the cardinality of the alphabet or symbols, respectively.

3.5.1. Comments on Abu Taha et al.'s scheme

Abu Taha*et al.* scheme is inefficient in hashing identical plaintext due to using the same key. On the other hand, Abu Taha*et al.* use matrix multiplication as the main operation similar to the original HC; it multiplies the plaintext with a non-invertible key matrix to calculate the hash message, which makes it vulnerable and weak when it is applied to the plaintext block containing zero values or a black color image. The output hash value will be similar to the plaintext (input value). The hashing output of the scheme will be similar to the tensor product scheme due to the matrix multiplication, and zero (black pixels) block input will be mapped to zero (black pixels) block.

4. Proposed Hash Function Based on Affine Hill Cipher HFB-AHC

The proposed hash algorithm for data integrity HFB-AHC varies from AHC in the following. To hash a plaintext P, the sender *A* and the receiver *B* securely share a non-invertible matrix, a secret SEED that is used to produce a pseudo-random sequence of permutations according to (14)

$$t_r = PRPermG(SEED, r) \tag{14}$$

obtaining the *r*-th output permutation from the pseudo-random permutation generator *PRPermG (*e.g., RC4 [10]). Permutes the rows and columns of a key matrix *K* according to t_r producing a new key-matrix $K_t=t_r(K)$. The number of dynamic keys used in the proposed hash function is *m!* where *m* denotes the block size. The hashing is then performed by (15)

$$H(P_i) = K_t \times P_i + V_i \, mod \, N \tag{15}$$

Where *H* represents the hash function, $H(P_i)$ is the output of the hashing algorithm, *K* is the key matrix, P_i is the plaintext block with index *i*, and $1 \leq i \leq n$, *n* is the number of blocks, V_1 is the initial vector generated randomly based on the pseudo-random number by using (16), $V_1=\{v_1, v_2, ...v_j\} \sqsubset Z_N-\{0\}$, $1 \leq j \leq m$, $Z_N=\{0, 1, 2, ..., N-1\}$. $V_2=H(P_1)$, $V_3=H(P_2)$, ...$V_i=H(P_{i-1})$.

$$V_1 \, (m) \tag{16}$$

Where PRSetGSEED(m) is a pseudo-random set sequence generator (e.g., RC4 initialized by SEED) returning

a set of cardinality m, m denoting the block size. It is worth mentioning that Period (RC4) is estimated to be greater than 10^{100} [10].

Example

Suppose that the (sender) A and the (receiver) B want to grantee the data integrity of the plaintext P_1=" 15 141 113 10 107 16 102 215" and P_2=" 65 148 132 0 19 61 229 136", note that N=256.

Our proposed HFB-AHC works as follows:

1. A and B exchange a secret non-invertible key matrix and a seed value to compute V_1 based on (16), V_1= {17, 12, 121, 139, 251, 223, 78, 151}, and suppose that

$$K = \begin{bmatrix} 14 & 2 & 19 & 221 & 231 & 119 & 24 & 67 \\ 10 & 72 & 194 & 36 & 119 & 121 & 22 & 13 \\ 20 & 15 & 250 & 243 & 127 & 75 & 70 & 84 \\ 121 & 15 & 250 & 243 & 127 & 75 & 20 & 84 \\ 79 & 19 & 7 & 1 & 124 & 247 & 110 & 88 \\ 192 & 250 & 87 & 71 & 105 & 35 & 6 & 75 \\ 128 & 235 & 70 & 52 & 119 & 23 & 0 & 7 \\ 1 & 45 & 60 & 78 & 20 & 49 & 9 & 39 \end{bmatrix}$$

   Note that K must be non-invertible; in the case of our example,
   The det(K) mod 256 = 224 ≠ 0, but gcd (256,224)=32. Thus K^{-1} does not exist over modulo N=256, and hence K is non-invertible.

2. The plaintext is divided into three blocks since,
   the size of K = 8, and the plaintext size = 16,
   the number of blocks = plaintext sizes/ key size = 16/8 = 2; in the case of plaintext size is not a multiple of key size, the last block will be padded by *x* values where *x* is calculated according to (17).

$$x = " ks \ - \ "("ps \ mod \ ks") \qquad (17)$$

   Where ks denotes the key size, and ps is the plaintext block size.

3. The plaintext block (i) will be hashed according to (15)
4. Set V_i=H(P_{i-1}), i≥2
5. Apply the permutation t_r over the key matrix K_t=t_r(K)
6. Repeat step 3 as long as i ≤ n (number of blocks).

The first block is hashed as follows:

$$H(P_1) = \begin{bmatrix} 14 & 2 & 19 & 221 & 231 & 119 & 24 & 67 \\ 10 & 72 & 194 & 36 & 119 & 121 & 22 & 13 \\ 20 & 15 & 250 & 243 & 127 & 75 & 70 & 84 \\ 121 & 15 & 250 & 243 & 127 & 75 & 20 & 84 \\ 79 & 19 & 7 & 1 & 124 & 247 & 110 & 88 \\ 192 & 250 & 87 & 71 & 105 & 35 & 6 & 75 \\ 128 & 235 & 70 & 52 & 119 & 23 & 0 & 7 \\ 1 & 45 & 60 & 78 & 20 & 49 & 9 & 39 \end{bmatrix} \times \begin{bmatrix} 15 \\ 141 \\ 113 \\ 10 \\ 107 \\ 16 \\ 102 \\ 215 \end{bmatrix} + \begin{bmatrix} 17 \\ 12 \\ 121 \\ 139 \\ 251 \\ 223 \\ 78 \\ 151 \end{bmatrix} mod\ 256$$

$$H(P_1) = \begin{bmatrix} 212 \\ 80 \\ 245 \\ 6 \\ 52 \\ 114 \\ 57 \\ 186 \end{bmatrix}$$

V_2=H(P_1)={212, 80, 245, 6, 52, 114, 57, 186}, suppose that the generated t_1={3, 1, 5, 6, 7, 2, 8, 4}, then after applying the permutation over the rows of the key matrix the following will be obtained:

$$K_{t_1} = \begin{bmatrix} 20 & 15 & 250 & 243 & 127 & 75 & 70 & 84 \\ 14 & 2 & 19 & 221 & 231 & 119 & 24 & 67 \\ 79 & 19 & 7 & 1 & 124 & 247 & 110 & 88 \\ 192 & 250 & 87 & 71 & 105 & 35 & 6 & 75 \\ 128 & 235 & 70 & 52 & 119 & 23 & 0 & 7 \\ 10 & 72 & 194 & 36 & 119 & 121 & 22 & 13 \\ 1 & 45 & 60 & 78 & 20 & 49 & 9 & 39 \\ 121 & 15 & 250 & 243 & 127 & 75 & 20 & 84 \end{bmatrix}$$

Then the second block is hashed by applying (15) by using the key (K_{t_1}) after applying the permutation:

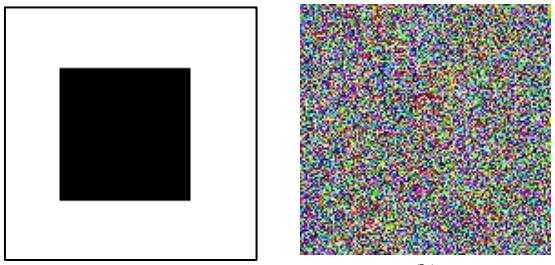
$$H(P_2) = \begin{bmatrix} 20 & 15 & 250 & 243 & 127 & 75 & 70 & 84 \\ 14 & 2 & 19 & 221 & 231 & 119 & 24 & 67 \\ 79 & 19 & 7 & 1 & 124 & 247 & 110 & 88 \\ 192 & 250 & 87 & 71 & 105 & 35 & 6 & 75 \\ 128 & 235 & 70 & 52 & 119 & 23 & 0 & 7 \\ 10 & 72 & 194 & 36 & 119 & 121 & 22 & 13 \\ 1 & 45 & 60 & 78 & 20 & 49 & 9 & 39 \\ 121 & 15 & 250 & 243 & 127 & 75 & 20 & 84 \end{bmatrix} \times \begin{bmatrix} 50 \\ 18 \\ 220 \\ 124 \\ 124 \\ 114 \\ 35 \\ 29 \end{bmatrix} + \begin{bmatrix} 212 \\ 80 \\ 245 \\ 6 \\ 52 \\ 114 \\ 57 \\ 186 \end{bmatrix} mod\ 2\,56$$

$$H(P_2) = \begin{bmatrix} 6 \\ 98 \\ 209 \\ 130 \\ 176 \\ 228 \\ 92 \\ 215 \end{bmatrix}$$

The given plaintext and the output of HFB-AHC (hashed value) will be sent by sender *A* to receiver *B*. The receiver B recomputed the hash value as the sender had done. The computed hash value HFB-AHC$_{computed_by_B}$(P) by the receiver B will be compared with the computed hash value HFB-AHC$_{computed_by_A}$(P) by sender A.

If HFB-AHC$_{computed_by_B}$(P) = HFB-AHC$_{computed_by_A}$(P), then the integrity is obtained.

On the other hand, we examined our proposed HFB-AHC scheme for hashing black background image; the visual inspection of the obtained results show that the proposed hash function is effective in hashing images with the black color area and with identical plaintext (see Figure 2).



(a) **(b)**

Fig. 2 Hashing a black background image. (a) Original image, (b) Hashed image HFB-AHC

5. Proposed Hash Function based on Tensor Product HFB-TP

The proposed hash function based on tensor product HFB-TP uses the same steps as in HFB-AHC but differs in constructing the key; it is well known that calculating the determinant of a matrix is considered a challenge for computation due to its complexity. Calculating the determinant of a matrix *n x n* requires n! steps produced through minors expansion. As a result, the number of steps required to compute the determinant of a given matrix of integer elements using minor expansion is bounded below by O(n!) [22]. On the other hand, if we use the Gaussian elimination method to calculate the determinant of a matrix, we need O(n^3) steps. The size of the matrix entries affects how many steps are necessary for practice. Calculating the determinants is necessary for the proposed scheme to decide whether the matrix is invertible or not invertible since we used a non-invertible matrix for hashing purposes. Constructing a non-invertible matrix with a large size is a challenge. Therefore the tensor product is used to construct the key; a small-size non-invertible matrix is used to construct/generate a large-size matrix. To hash a plaintext P, the sender *A* and the receiver *B* securely share a non-invertible matrix K and calculate the key matrix K_h according to (18).

$$K_h = K \otimes K^T \otimes K \qquad (18)$$

Based on (18), the size of the generated matrix is n^3; for example, if n=2, then the size of the generated matrix is 8. HFB-TP follows the same steps as in HFB-AHC except in the key construction. The key construction of the proposed HFB-TP is depicted and illustrated by the following example:

Suppose, $K = \begin{bmatrix} 27 & 14 \\ 95 & 4 \end{bmatrix}$, then $K^T = \begin{bmatrix} 27 & 95 \\ 14 & 4 \end{bmatrix}$, det(k) mod 256=58≠0 but gcd(58,256)=2, thus *K*is a non-invertible matrix. The result of the tensor product of non-invertible matrices is a non-invertible matrix. K_h can be calculated and constructed according to (18) as follows:

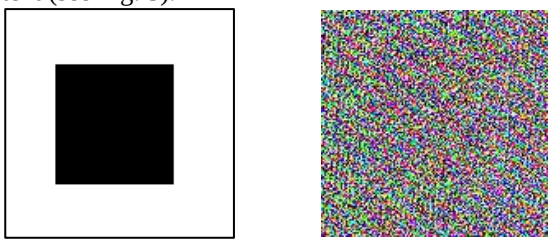
$$K_{temp} = K \otimes K^T = \begin{bmatrix} 27 & 14 \\ 95 & 4 \end{bmatrix} \otimes \begin{bmatrix} 27 & 95 \\ 14 & 4 \end{bmatrix} mod\ 2\,56$$

$$= \begin{bmatrix} 217 & 5 & 122 & 50 \\ 122 & 108 & 196 & 56 \\ 5 & 65 & 108 & 124 \\ 50 & 124 & 56 & 16 \end{bmatrix}$$

$$K_h = K_{temp} \otimes K = \begin{bmatrix} 217 & 5 & 122 & 50 \\ 122 & 108 & 196 & 56 \\ 5 & 65 & 108 & 124 \\ 50 & 124 & 56 & 16 \end{bmatrix} \otimes \begin{bmatrix} 27 & 14 \\ 95 & 4 \end{bmatrix}$$

$$K_h = \begin{bmatrix} 227 & 222 & 135 & 70 & 222 & 172 & 70 & 188 \\ 135 & 100 & 219 & 20 & 70 & 232 & 142 & 200 \\ 222 & 172 & 100 & 232 & 172 & 184 & 232 & 16 \\ 70 & 232 & 20 & 176 & 188 & 16 & 200 & 224 \\ 135 & 70 & 219 & 142 & 100 & 232 & 20 & 200 \\ 219 & 20 & 31 & 4 & 20 & 176 & 4 & 240 \\ 70 & 188 & 20 & 200 & 232 & 16 & 176 & 224 \\ 142 & 200 & 4 & 240 & 200 & 224 & 240 & 64 \end{bmatrix}$$

On the other hand, we examined our proposed hash function based on tensor product HFB-TP for hashing black background image; the visual inspection of the obtained results show that the proposed HFB-TP is effective in hashing images with the black color area and with identical plaintext (see Fig. 3).



(a) (b)

Fig. 3 Hashing a black background image. (a) original image, (b) Hashed image HFB-TP

6. Proof of the Proposed HFB-AHC and HFB-TP Hash Functions

In this section, we discuss the former mentioned four requirements in section 2 for having an effective hash function. We prove that our proposed HFB-AHC and HFB-TP satisfy those requirements.

6.1. Applicable to any Arbitrary Size of Data

The proposed HFB-AHC and HFB-TP hash functions effectively hash any arbitrary data size; it follows the same steps as in HC; more precisely, if the plaintext size is less than the key size, then the plaintext will be padded with (key size – plaintext size). In the case of plaintext size is a multiple of key size, then it is divided into a number of blocks plaintext size div key size; finally, in the case of plaintext size is not a multiple of key size and it is greater than the key size, the plaintext is divided into many blocks, and the last block will be padded by x values where x is calculated according to (13).

6.2. Fixed Size Output

The proposed HFB-AHC and HFB-TP hash functions can process any arbitrary size. This requirement has been proofed in the given an example (section 4). The proposed HFB-AHC and HFB-TP effectively hashed a sequence of blocks and produced a fixed data length. Figure 2 and Figure 3 showed that HFB-AHC and HFB-TP effectively hashed images with a large black background area.

6.3. Easy to Compute

The hashing process is performed by multiplying the plaintext P, with a key matrix and carrying out an addition over modulo N; the proposed HFB-AHC and HFB-TP have the same operation as in AHC. It has only two primitive operations (matrix multiplication and addition over modulo N).

6.4. One Way Property

Our proposed HFB-AHC and HFB-TP hash algorithms used a non-invertible matrix. Hence the produced hashed value cannot be decrypted due to the fact that K^{-1} does not exist, and hence the hashed value cannot be reversed back. Hence the one-way property is satisfied.

7. Security Analysis

A good indicator of a cryptosystem's performance is its ability to tolerate and resist various cryptanalysis and attacks [23]. The security of our hashing algorithm is assessed using its robustness against attacks. From a strongly cryptographic point of view, it is demonstrated that our suggested hash HFB-AHC and HFB-TP are secure. As described and addressed in the following subsections.

7.1. Known Plaintext-Ciphertext Attack KPCA

The KPCA is effective if we use the same invertible key matrix. Our proposed HFB-AHC and HFB-TP hash functions use a non-invertible matrix; hence the KPCA does not apply to our proposed HFB-AHC and HFB-TP, and thus it is secure against the KPCA.

7.2. Key Space Analysis

The complete variety of keys available for use in cryptosystems is known as the key space. The key space must be sufficiently large to prevent brute force attempts from being considered to be secure. For the HFB-AHC and HFB-TP, the key space is the same as that of HC [11], [12]. In fact, it covers all matrices that are not invertible. Since we utilized a non-invertible matrix, the key matrix can also be a rectangular matrix. Therefore the key space of the proposed HFB-AHC and HFB-TP is large; hence it is secure against brute-force attacks.

8. Conclusion

Information security heavily depends on cryptographic hash functions, a core of cryptosystems. It aims to provide non-repudiation, confidentiality, authentication, and integrity. Hash functions must therefore be robust and effective, as demonstrated by the significance of hash functions and their use in different applications. Thus far, we proposed novel hash functions for data integrity based on the affine Hill cipher and tensor product; the proposed HFB-

AHC and HFB-TP are secure and effective in hashing the plaintext blocks; this is very clear when handling and hashing black or zero plaintext blocks. We proved that HFB-AHC and HFB-TP satisfy the hash algorithm requirements. The security analysis showed that the proposed HFB-AHC and HFB-TP are secure against the KPCA and brute force attacks.

References

[1] M. Farajallah, M. Abu Taha, and R. Tahboub, "A Practical One Way Hash Algorithm Based on Matrix Multiplication," In *International Journal of Computer Applications*, vol. 23, no. 2, pp. 34–38, 2011. Crossref, http://doi.org/10.5120/2859-3677

[2] Hill L. S, "Cryptography in an Algebraic Alphabet," *American Mathematical*, vol. 36, no. 6, pp. 306–312, 1929. Crossref, https://doi.org/10.1080/00029890.1929.11986963

[3] Hill L. S, "Concerning Certain Linear Transformation Apparatus of Cryptography," *The American Mathematical Monthly*, vol. 38, no. 3, pp. 135–154, 1931. Crossref, https://doi.org/10.2307/2300969

[4] A. G. Mahmoud, A. Y, and Chefranov, "Hill Cipher Modification Based on Eigenvalues HCM-EE," *Proceedings 2nd International Conference on Security of Information and Networks*, pp. 164–167, 2009. Crossref, https://doi.org/10.1145/1626195.1626237

[5] A. G. Mahmoud, A. Y, and Chefranov, "Secure Hill Cipher Modifications and Key Exchange Protocol," *2010 IEEE International Conference on Automation, Quality and Testing, Robotics*, vol. 2, pp. 1–6, 2010. Crossref, https://doi.org/10.1109/AQTR.2010.5520828

[6] A. Mahmoud and A. Chefranov, "Hill Cipher Modification Based on Pseudo-Random Eigenvalues," *Applied Mathematics and Information Sciences*, vol. 8, no. 2, pp. 505–516, 2014. Crossref, https://doi.org/10.12785/Amis/080208

[7] A. Y. Mahmoud and A. G. Chefranov, "A Hill Cipher Modification Based on Eigenvalues Extension with Dynamic Key Size HCM-EXDKS," *International Journal of Communication Networks and Information Security*, vol. 6, no. 5, pp. 57–65, 2014. Crossref, https://doi.org/10.5815/Ijcnis.2014.05.08

[8] A. Y. Mahmoud and M. M. Abu-Saqer, "Modification of Select Operation Model for Multilevel Security: Medical Database Systems as an Application," *International Conference on Assistive and Rehabilitation Technologies 2020 Icare Tech 2020*, pp. 47–50, 2020. Crossref, https://doi.org/10.1109/Icaretech49914.2020.00016

[9] W. A. Kiele, "A Tensor-Theoretic Enhancement to the Hill Cipher System," *Cryptologia*, vol. 14, no. 3, pp. 225–233, 1990. Crossref, https://doi.org/10.1080/0161-119091864931

[10] W. Stallings, "*Cryptography and Network Security Principles and Practices*," 2012.

[11] S. Saeednia, "How to Make the Hill Cipher Secure," *Cryptologia*, vol. 24, no. 4, pp. 353–360, 2000. Crossref, https://doi.org/10.1080/01611190008984253

[12] J. Overbey, W. Traves, and J. Wojdylo, "On the Keyspace of the Hill Cipher," *Cryptologia*, vol. 29, no. 1, pp. 59–72, 2005. Crossref, https://doi.org/10.1080/0161-110591893771

[13] Chaitra D B, Dr. Rashmi R Rachh, "Lightweight Integrity Verification in Named Data Networking," *SSRG International Journal of Computer Science and Engineering*, vol. 4, no. 8, pp. 5-10, 2017. Crossref, https://doi.org/10.14445/23488387/IJCSE-V4I8P102

[14] M. Selvavathi, S.Edwin Raja, "Anticipation of Vulnerable Attacks in Vanet Using Blockchain Technique," *SSRG International Journal of Computer Science and Engineering*, vol. 8, no. 1, pp. 19-23, 2021. Crossref, https://doi.org/10.14445/23488387/IJCSE-V8I1P104.

[15] A. Shamir, "New Directions in Cryptography," *Lecture Notes in Computer Science, (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2162, pp. 159, 2001. Crossref, https://doi.org/10.1007/3-540-44709-1_14

[16] A. Y. Mahmoud, "*Development of Matrix Cipher Modifications and Key Exchange Protocol*," Doctor of Philosophy in Computer Engineering. Thesis (Ph.D.)-Eastern Mediterranean University, Faculty of Engineering, Dept. of Computer Engineering, 2012.

[17] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive Study of Symmetric Key and Asymmetric Key Encryption Algorithms," *2017 International Conference on Engineering and Technology, ICET 2017*, pp. 1–7, 2017. Crossref, https://doi.org/10.1109/Icengtechnol.2017.8308215

[18] M. Gupta, S. Mahto, and A. Patel, "Implementation of 128, 192 & 256 Bits Advanced Encryption Standard on Reconfigurable Logic," *International Journal of Engineering Trends and Technology*, vol. 50, no. 6, pp. 305–309, 2017. Crossref, https://doi.org/10.14445/22315381/IJETT-V50P251

[19] Xiao Luo, Haixin Wang, Daqing Wu, Chong Chen, Minghua Deng, Jianqiang Huang, Xian-Sheng Hua, "A Survey on Deep Hashing Methods," *The ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 2022. Crossref, https://doi.org/10.1145/3532624.

[20] S. Han, K. Xu, Z. Zhu, S. Guo, H. Liu, and Z. Li, "Hash-Based Signature for Flexibility Authentication of IOT Devices," *Wuhan University Journal of Natural Sciences*, vol. 27, no. 1, pp. 1–10, 2022. Crossref, https://doi.org/10.1051/Wujns/2022271001

[21] M. Phys, E. U. Moya-S, and E. Bayro-Corrochano, "On a Tomic Functions for Image," pp. 1–32, 2012.

[22] Gilbert Strang, "*Introduction to Linear Algebra*," 5th Ed., Wellesley-Cambridge Press, 2016.

[23] H. E. D. H. Ahmed, H. M. Kalash, and O. S. Farag Allah, "An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) for Image Encryption and Decryption," *Informatica*, vol. 31, no. 1, pp. 121–129, 2007.

[24] Dr. M E Purushoththaman, Dr. Bhavani Buthtkuri, "Effective Multiple Verification Process Ensuring Security and Data Accuracy in Cloud Environment Storage," SSRG International Journal of Computer Science and Engineering, vol. 6, no. 7, pp. 1-4, 2019. Crossref, https://doi.org/10.14445/23488387/IJCSE-V6I7P101

[25] T. H. Bar, *Invitation to Cryptology*, 2002.

[26] M. Mokhtari and H. Naraghi, "Analysis and Design of Affine and Hill Cipher," *Journal of Mathematics Research*, vol. 4, no. 1, pp. 67-77, 2012. Crossref, https://doi.org/10.5539/Jmr.V4n1p67