Key Schedule Algorithm using 3-Dimensional Hybrid Cubes for Block Cipher

Muhammad Faheem Mushtaq[1], Sapiee Jamel[2], Siti Radhiah B. Megat[3], Urooj Akram[4], Mustafa Mat Deris[5]
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, 86400 Batu Pahat, Johor, Malaysia[1, 2, 3, 5]
Faculty of Computer Science and Information Technology
Khwaja Fareed University of Engineering and Information Technology, 64200 Rahim Yar Khan, Pakistan[1, 4]

***Abstract*—A key scheduling algorithm is the mechanism that generates and schedules all session-keys for the encryption and decryption process. The key space of conventional key schedule algorithm using the 2D hybrid cubes is not enough to resist attacks and could easily be exploited. In this regard, this research proposed a new Key Schedule Algorithm based on coordinate geometry of a Hybrid Cube (KSAHC) using Triangular Coordinate Extraction (TCE) technique and 3-Dimensional (3D) rotation of Hybrid Cube surface (HCs) for the block cipher to achieve large key space that are more applicable to resist any attack on the secret key. The strength of the keys and ciphertext are tested using the Hybrid Cube Encryption Algorithm (HiSea) based on Brute Force, entropy, correlation assessment, avalanche effect and NIST randomness test suit which proves the proposed algorithm is suitable for the block cipher. The results show that the proposed KSAHC algorithm has performed better than existing algorithms and we remark that our proposed model may find potential applications in information security systems.**

Keywords*—Encryption; decryption; key schedule algorithm; hybrid cube; block cipher

I. Introduction

Security is the major concern due to the fast growth of the internet in today's digital world and it is important to provide security of data from unauthorized access [1]. Hence, secure communication is the basic requirement of every transaction over networks. Cryptography is an important component to ensure secure communication of data by using the security services like confidentiality, authentication, data integrity and non-repudiation. Data confidentiality refers to the protection of sensitive data from being accessed by unauthorized parties [2]. Traditionally, the cryptographic algorithms comprise of different mathematical and logical components integrated together as part of the algorithms [3], [4], [5]. The development of the fully secured cryptographic algorithm is difficult due to the challenges from cryptanalysts who continuously trying to break any available cryptographic systems. So, the selection of the right cryptographic algorithm is essential to accomplish the high-security requirements to ensure the protection of cryptographic components from cryptanalysis [6], [7]. In this regard, a key schedule algorithm is employed to generate secret keys and plays an important role in the development of encryption schemes. In order to resist the related key attack, many types of research were conducted to develop a powerful and significant key generation algorithm that increases the difficulty for a cryptanalyst to recover the secret key [8]. All cryptographic algorithms are recommended to follow 128, 192 and 256-bits key lengths proposed by Advanced Encryption Standard (AES) [9].

Permutation plays an important role in the development of cryptographic algorithms and it contains the finite set of numbers or symbols that are used to mix up a readable message into ciphertext as shown in transposition cipher [10]. The logic behind any cryptographic algorithm is the number of possible combinations in the key space, and bigger key-space could be achieved from the used of flips and twists of the elements of the cube which ensure every state of the cube is actually permuted [11], [12]. An image permutation algorithm based on the geometrical projection and shuffling in the design of key schedule algorithm is used to increases the security of original image by preventing it from outside attacks [13], [14]. The computing information about complex geometric primitives is often costly, while computational geometry determines many asymptotically effective algorithms for such problems are indicated [15]. It mentioned that if the coordinate system is adopted in the plane then key quantities can be circularly permuted. The rotation of the object is used to shift the position along a circular path in the *xy* plane. Whereas, the translation of an object is employed to shift the position along the straight path from one coordinate to another [16], [17]. The construction of magic cubes using the concept of a magic square and two orthogonal Latin squares was proposed [18]. The magic cube is 3-Dimensional (3D) coordinates consisting of six faces that are used in the development of complex permutation as apart of the design of cipher. The cubes rotation technique is applied to the image pixels to produce an encrypted picture [19] and revert the rotation to decrypt the image. Moreover, the image scrambling technique using the rotation of rows and columns of the magic cube is used to break the relationship between the image elements which creates the encryption [20]. Similarly, the cube rotation technique has been applied to encrypt text which provides more security and efficiency [21] compared to other operations.

Hybrid cubes are generated on the basis of Latin squares, Orthogonal Latin Squares, Magic Squares and Magic Cubes [7]. Based on that, a new way was found for further development of new transformation based on a permutation of integer numbers and develops a non-binary block cipher. Furthermore, a non-binary block cipher is proposed using all possible combination of 2-Dimensional (2D) hybrid cubes as the source for the encryption and decryption keys [22]. The Rajavel's encryption algorithm and the cubical key are generated using the hybridization and rotation of hybrid cubes by shuffling the cubes [23]. Similarly, the HiSea encryption algorithm performed hybridization with 2D hybrid cubes which are a very time-consuming process and it generates the key space that is not sufficient to resist attacks and could easily be exploited. This research opens a new way for creating a key schedule algorithm using 3D hybrid cubes based on permutation and combination of integer numbers.

In this regard, this research proposed a new Key Schedule Algorithm based on the coordinate geometry of a Hybrid Cube (KSAHC) for HiSea encryption algorithm [22]. The KSAHC transformation using Three-Dimensional (3D) hybrid cube is to create a large key space that will be used as encryption and decryption keys which makes the Brute Force attack difficult and time-consuming. KSAHC encryption keys are represented as an $n \times n \times n$ matrix of integer numbers and used in the development of the permutation and substitution of order 4 square matrix. A new cube structure based on the Triangular Coordinate Extraction [24] and rotation of Hybrid Cube surface [25] is proposed where the layer entries are between a set of integers from 1618 to 11198. TCE technique is used to extract the coordinate of hybrid cube during the rotation and it plays an important role in the development of KSAHC algorithm. Also, this transformation will be able to generate random ($4 \times 4 \times 4$) matrices from any hybrid cube layers. Furthermore, the proposed algorithm has been implemented into existing non-binary block cipher and observes the promising impact of keys on the ciphertext. Hence, by using the proposed KSAHC algorithm for HiSea, this research leads to the enhancement of the strength and validation of encryption key and the cipher. The contributions and novelty of this study are as follows:

- To propose a new Key Schedule Algorithm based on coordinate geometry of a Hybrid Cubes (KSAHC) for the non-binary block cipher.
- To overcome the problem of small key space that could occur due to the 2D hybrid cube layers, a new approach has been adopted based on the 3D rotation of HCs by using the columns and rows shift transformation.
- A comparative analysis of the proposed algorithm with AES, HiSea, and DKSA has been completed, and the strength of the encryption keys and ciphertext are examined based on Brute Force, entropy, correlation assessment, avalanche effect, and NIST randomness test suit.
- The novelty of this research is to incorporate the concept of coordinate geometry, 3D rotation of *HCs* and TCE technique into the key scheduling algorithm.

The remaining paper is organized as follows: Section II discusses the material and methods in which explain the detail design of the proposed Key Schedule Algorithm based on coordinate geometry of Hybrid Cube (KSAHC) for the non-binary block cipher. Section III explains the results and discussion of the proposed algorithm. Section IV presents the conclusion and future work of this research.

II. Material and Methods

This section discusses the design of the new Key Schedule Algorithm based on coordinate geometry of a Hybrid Cubes (KSAHC) for HiSea encryption algorithm [22].

The schematics of a conventional HiSea Encryption algorithm uses individual cubes with a 2D structure for the development of key schedule algorithm but the proposed KSAHC algorithm has been developed using the 3D structure of hybrid cube and its rotation using ShiftColumn and ShiftRow transformation as shown in Fig. 1. The first step in the development of KSAHC is to generate rotation of *HCs* which is the main element of the construction of encryption and decryption key in the key schedule algorithm. The shuffling and mixing of rows and columns of hybrid cube provide the resultant matrices that must be invertible. These invertible matrices are used in the development of encryption and decryption keys in the symmetric block cipher. The second step develops a key scheduling algorithm which involves the generation of the key table using the matrices that are utilized in the rotation of the hybrid cube. Previously, the inner matrix multiplication of magic cubes is used to generates hybrid cubes and these cubes are used to generate a key table in the key schedule algorithm. The third step involves encryption of messages using the encryption key to form ciphertext that comprises of integer numbers string employed in the construction of non-binary block cipher and final step is to decrypt the message to form an encryption algorithm. For this purpose, the detailed design of the key schedule algorithm using the triangular 3D rotation of *HCs*, HiSea encryption, and decryption algorithms are discussed. Based on Fig. 1, the entire elements in the dotted box illustrate the components of KSAHC framework. The rotation is needed in transforming the ciphertext to plaintext to obtain the original plaintext from the ciphertext.

A. Proposed KSAHC Algorithm

The proposed key schedule algorithm undergoes through the following main phases; generation of key matrices using TCE technique, ShiftColumns, ShiftRows, unique matrices operation, and triangular key matrices. Using the layers of hybrid cubes [22], a new cube structure is generated that are used to computationally secures the encryption and decryption process of existing HiSea encryption algorithm. Every algorithm is almost having inputs, processes, and output. The input to the proposed key schedule algorithm is the user password. The KSAHC algorithm determines a permutation based on user password in generating the triangular key matrices from all faces of the hybrid cube. The Initial Vector (IV) is also used as the secondary security measure with the user password to prevent repetition in keys. The process of 3D rotation of *HCs* can be explained in Fig. 2.

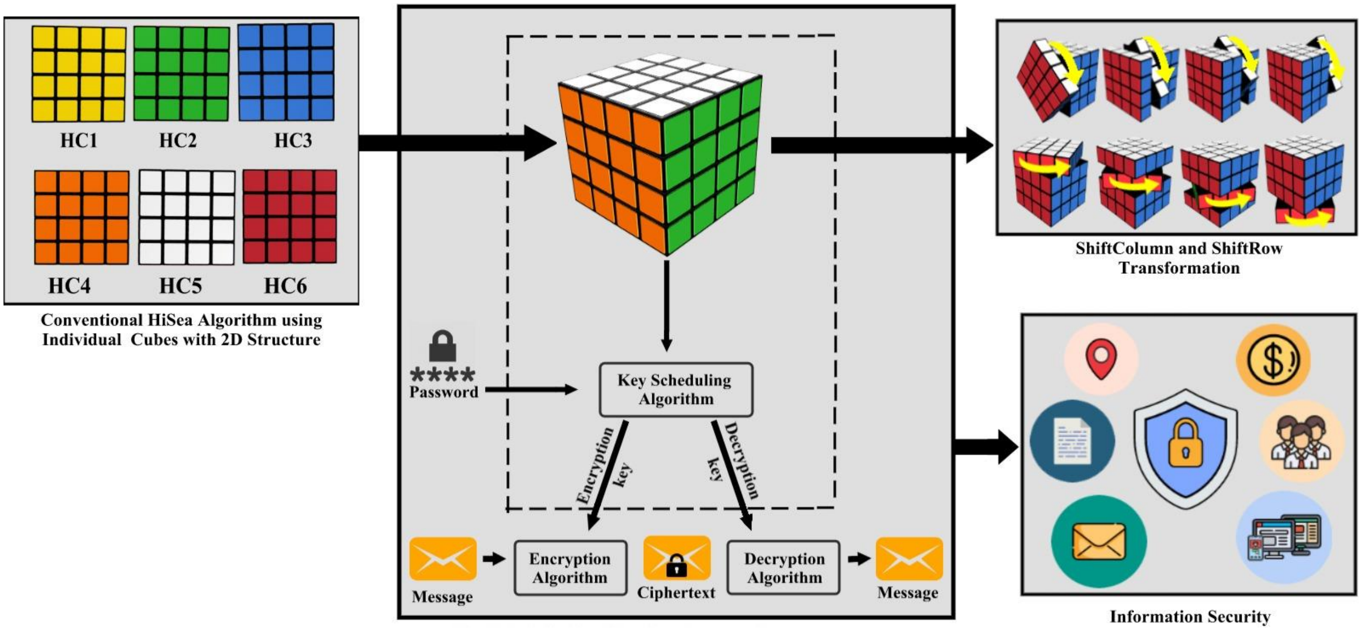

Fig. 1. Framework of the Proposed KSAHC Algorithm.

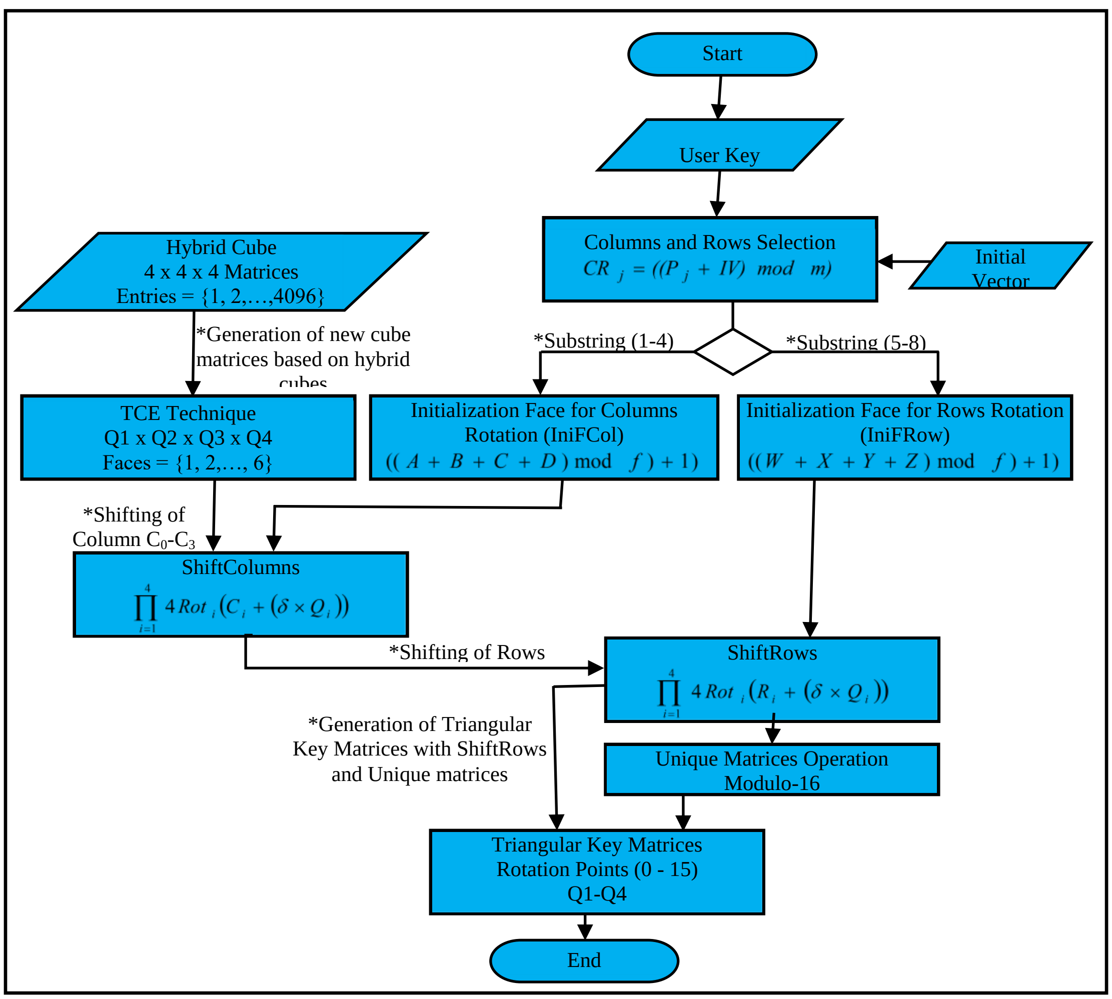

Fig. 2. The Process of KSAHC Algorithm.

The 8-digit value of the selection of Columns and Rows (CR_j) is employed for the rotation of columns and rows of the hybrid cubes. Furthermore, the CR_j is also used to identify the initialization faces of columns and rows. ShiftColumns and ShiftRows operations are performed on different faces of the hybrid cube and TCE technique is demonstrated. The 3D rotation pattern of both operations is depending on the CR_j value. A new combination of triangular key matrices is generated by using the ShiftRows operation and the rotation points based on the unique matrices. This new combination of layer entries using the rotation of *HCs* can be used to add randomness in the output of the proposed algorithm. Only the invertible triangular key matrices of the hybrid cube are used as the encryption and decryption keys in the non-binary block cipher. Furthermore, the encryption and decryption algorithm of HiSea encryption algorithm is also discussed in order to test the strength of the proposed algorithm. The detailed design of these components will be explained in the following sub-section.

1) Generation of key matrices: The input key matrices (also referred to as the states) are structured into the 4 × 4 matrices. In this regard, 24 key matrices are generated using the HiSea encryption algorithm [22] and the TCE technique is demonstrated. By using the TCE technique, each key matrix from HiSea key table is used to generate one row in the new cube matrix. Similarly, four key matrices are used to generate one face of the hybrid cube. Finally, 24 key matrices are employed to generate six key matrices that are used in the rotation of HCs as shown in Fig. 3.

a) Design of Triangular Coordinate Extraction (TCE) Technique: The design of HCs is divided into six faces and each face is further divided into four quarters (Q1, Q2, Q3, Q4) by intersection of two diagonal lines pass through the center of a circle [24]. The primary diagonal lies on the x-axis while the secondary diagonal is on the y-axis as presented in Fig. 4. The element of the coordinates of the hybrid cube shown as f, i and j in which the element f shows the six faces of the hybrid cube, while i represents the rows and the j represents the columns of the hybrid cube. Firstly, the rotation of HCs of face 1 is considered and after that, a similar process is applied in the other faces. The rotation of triangular HCs is counterclockwise which is the main component in the generation of the encryption keys. The rotation points 0 to 15 around the quarters Q1 to Q4 represent the position of coordinates in which the Q1 is the passage from rotation points 0 to 4, Q2 lies between 4 to 8, Q3 lies from 8 to 12 and Q4 is the passage from 12 to 0.

Definition 1. Let the *HCs* be $4 \times 4 \times 4$ square matrices, then the center of *HCs* is calculated by using the intersection of primary and secondary diagonals and it is possible if the elements of coordinates of rows and columns satisfy the reflexive and symmetric properties [25]. The properties of diagonals for the six faces of the hybrid cube are as follows:

2) Primary diagonal is defined as the collection of entries *HCs* (f, i, j), where $i = j$. The coordinates of primary diagonal for each face (f) of hybrid cube comprises the follows:

{(1, 1), (2, 2), (3, 3), (4, 4)}

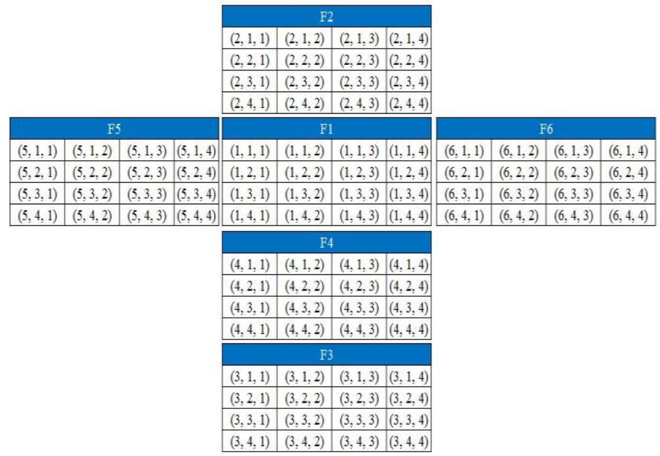

Fig. 3. Generation of Key Matrices.

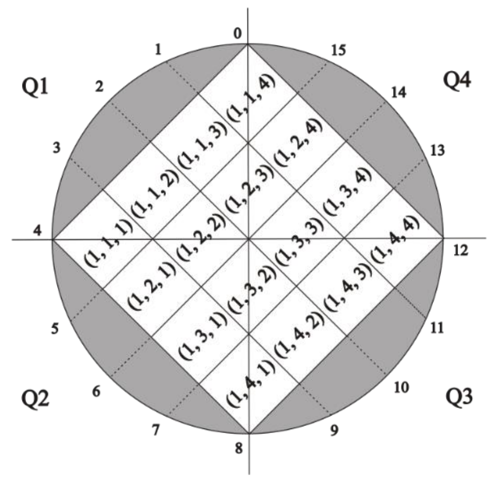

Fig. 4. Design of TCE Algorithm.

3) A secondary diagonal is defined as the collection of entries *HCs*(f, i, j), where $i + j = 5$ that can be calculated by the sum of the mean of the symmetric coordinates *(i, j)* and *(j, i)* of *HCs* matrix [24]. The coordinates comprise the secondary diagonal for each face (f) of the hybrid cube, as follows:

{(1, 4), (2, 3), (3, 2), (4, 1)}

When the value of diagonals *HCs*(f, i, j) satisfy the properties of primary and secondary diagonals then the value of coordinates of a particular cell is 1/2*HCs(f, i, j).*

The quarters *Q1* to *Q4* is used to extract the coordinates value during the rotation of *HCs* and shifting the value from quarters *Q1* to *Q4* based on properties discussed in Definition 1. The value of the coordinates of *HCs* can be extracted based on the following equations in Table I.

In this technique, each quarter of the square matrix is used to generate the one coordinate for a newly generated key matrix based on the quarter's equations. Similarly, the quarters *Q1* to *Q4* are used to generate one row for the new key matrix. In this regard, four key matrices are required to generate the order 4 key matrix. Furthermore, the new key matrix is rotated from quarters *Q1* to *Q4* and the value of the coordinates are shifted according to the rotation pattern.

TABLE. I. COORDINATES EXTRACTION FROM Q1 TO Q4

Quarters	Extraction of coordinates value
Q1	$\sum_{i=0}^{1}\sum_{j=1+i}^{4-i}(i+1, j)$
Q2	$\sum_{j=0}^{1}\sum_{i=1+j}^{4-j}(i, j+1)$
Q3	$\sum_{i=0}^{1}\sum_{j=1+i}^{4-i}(4-i, j)$
Q4	$\sum_{j=0}^{1}\sum_{i=1+j}^{4-j}(i, 4-j)$

4) Columns and rows selection: Definition 2. Let CR_j be the determinant of columns and rows rotation, when

$$CR_j = (P_j + IV)\, mod\ m \tag{1}$$

where, CR_j is represented as the resultant value of columns and rows, denoted as ABCDWXYZ, P_j is represented as Password, IV is the initialization vector, and m is the string of 8-integer numbers.

The input of the KSAHC algorithm can be considered as N. The m and IV are considered as the pre-defined strings of an integer number and the P_j is also obtained from the user. The overall input is the addition of IV and P_j (j = 0, 1, 2, …, N) that result is modulo with respect to m which is denoted as CR_j and can be represented in Equation (2).

$$CR_j = \begin{cases} P_j & if \quad 0 \le i \le N \\ IV & if \quad j = N \in Z^+ \\ m & if \quad j = N \bmod 12345678 \end{cases} \tag{2}$$

The resultant value ABCDWXYZ of CR_j is employed for the rotation of columns and rows. The value of *ABCD* shows the rotation of columns in which the first column (C0) is rotated based on *A*, second column (C1) is rotated based on *B*, third column (C2) is rotated based on *C*, and fourth column (C3) is rotated based on *D*. Whereas, the value of WXYZ shows the rotation of rows in which first row (R0) is rotated based on W, second row (R1) is rotated based on *X,* third row (R2) is rotated based on *Y*, and fourth row (R3) is rotated based on *Z*. Also, the *IV* is a random number that is used with the password in the key schedule algorithm and it is used only one time in each session. The purpose of using *IV* is to prevent repetition in keys, which make it difficult for the cryptanalysis to find the keys pattern and break the cipher.

5) Initialization face of columns rotation: Definition 3. Let *A*, *B*, *C* and *D* be the value of hybrid cube that is obtained from the resultant value of CR_j and it is used for the initialization of column rotation. It is also defined as the first four bits of CR_j that modulo with the hybrid cube faces.

$$\text{IniFCol} = (((A + B + C + D) \bmod f) + 1) \tag{3}$$

where, *IniFCol* represent the initialization face for columns rotation and *f* represent the six faces of hybrid cube.

6) Initialization face of rows rotation: Definition 4. Let *W*, *X*, *Y* and *Z* be the value of hybrid cube that is taken from the resultant value of rows and columns selection and it is employed for the initialization of row rotation. It is also defined as the last four numbers of the value of columns and rows selection.

$$\text{IniFRow} = (((W + X + Y + Z) \bmod f) + 1) \tag{4}$$

where *IniFRow* represent the initialization face for rows rotation and *f* represent the six faces of the hybrid cube.

7) The shiftcolumns transformation: As indicated by its name, the ShiftColumns transformation processes different columns between different faces of the hybrid cube. The operation of shifting the columns of the cube states over the specified column offsets is denoted as:

ShiftColumns(States)

The 3D rotation of columns of each face of order 4 matrix with other faces of hybrid cubes depends on first four value of the CR_j. For example, the column vector C_0 face 1 matrix is shifted over the C_0 vector of face 2. The C_0 vector of face 2 is shifted over the C_0 vector of face 3, C_0 vector of face 3 over the face 4, C_0 vector of face 4 is shifted over the face 1 and the C_0 vector of face 1 is shifted over the face 2. Similarly, the columns C_1, C_2 and C_3 of each face have shifted the coordinates of the selected columns. The ShiftColumns operations can be performed on different faces of the hybrid cube, so the rotation pattern of shifting columns of different faces depends on the CR_j value.

Definition 5. Let the ShiftColumns operation be the transposition of column vectors that cyclically shifts the columns of each face over the different column offsets of the hybrid cube. If the different faces of a cube having first and fourth columns then the value of $\delta = 1$, otherwise the value of the middle column is $\delta = 0$ as shown in Equation (5).

The processing equation of ShiftColumns is computed as follows:

$$ShiftColumns = \prod_{i=1}^{4} 4Rot_i\left(C_i + \left(\delta \times Q_i\right)\right) \tag{5}$$

where Rot_i is represented as the number of rotations based on the CR_j value, Ci is the column vectors of cube faces and the Q_i is the rotation of quarters Q_1 to Q_4.

Equation (5) defines each rotation Rot_i based on the CR_j value that affects the changes of column vectors rotation into 4 times on different faces of the hybrid cube because the ShiftColumns transformation is applied on four faces of the cube. Similarly, if the CR_j value is two then it affects the column vectors 8 times, if the CR_j value is three then it affects the column vectors 12 times, if CR_j value is four then it affects the column vectors 16 times, and so on. As mentioned earlier, the rotation pattern of columns depends on the first four (*ABCD*) values of CR_j that rotates the respective columns (C_0, C_1, C_2 and C_3) shown in Fig. 5.

Fig. 5. ShiftColumns Transformation.

For example, the column C_0 is rotated based on the value of A. Suppose the value of A is 3 then the column vector C_0 is rotated 3 times to different faces of the cube. Similarly, the value of B is responsible for shifting the coordinates of respective column vector C_1, the value of C shifts the coordinates of column C_2 and the value of D shifts the coordinates of respective column C_3. Moreover, if the rotation Rot_i of the cube is having the sided columns (C_0, C_3) where the $\delta = 1$ that also affects both sides faces of the cube and it rotates using the triangular coordinate extraction technique of quarters rotation Q_i. The extraction of the coordinate's value during the rotation of *HCs* and shifting the value from quarters Q_1 to Q_4 based on the rotation pattern. The rotation of quarters depends on the left-sided column (C_0) and right-sided column (C_3) of the hybrid cube. There are two different approaches used in the quarter's rotation of hybrid cube which is clockwise and counterclockwise. Firstly, if the rotation is based on C_0 then the quarters are being rotated to counterclockwise and the number of rotations depends on the Rot_i value. So, the process of a single rotation of column is that the value of Q_1 is shifted to Q_2, Q_2 value shifted to Q_3, Q_3 value shifted to Q_4 and finally Q_4 to Q_1. Similarly, if the rotation is based on C_3 then the quarters will be rotated towards the clockwise direction and the process of a single rotation of column is that the value of Q_1 is shifted to Q_4, Q_4 is shifted to Q_3, Q_3 to Q_2 and Q_2 is finally shifted to Q_1. The number of rotation of quarters Q_i depends on the value of Rot_i in the CR_j.

For ShiftColumns transformation, various permutations are performed on the column to add confusion in the key matrices of the hybrid cube. The rotation pattern of different faces can be divided into two different cases and the pseudocode for both cases of ShiftColumns transformation used to rotate the *HCs* is shown in Algorithm 1.

a) Case 1: IniFCol value is between 1 TO 4: If the IniFCol value is between 1 to 4, then we employ the rotation pattern of column vectors of F1, F2, F3 and F4 faces. The rotation can be performed counterclockwise direction and the rotation of columns of each face affects the coordinate value of other cube faces. Each time the rotation of C0 of each face affects the rotation of quarters of face F5 and it rotates to counterclockwise using the TCE technique. Also, each time rotation of C3 in each face affects the face F6 and it rotates to clockwise using TCE technique.

b) Case 2: IniFCol value is 5 or 6: If the *IniFCol* value is 5 or 6, then the column rotation uses the *F2*, *F4*, *F5* and *F6* faces. The rotation of column 1 and column 4 affects face 3 and 1, and these faces rotate counterclockwise and clockwise, respectively with the TCE technique.

Algorithm 1. Pseudo-Code of ShiftColumn operation

Assign suitable values to Rot_i, CR_j
loop
Calculate ShiftColumns where case 1 and 2 are calculated using Equation 5.
for the hybrid cube columns C_0 to C_3 **do {**
  if the hybrid cube column C_0 OR C_3 **then**
   for number of rotations Rot_i based on CR_j value **do**
    if the hybrid cube column C_0
     LeftFace ← RotateMatrixCounterClockWise(LeftFace)
    else
     RightFace ← RotateMatrixClockWise(RightFace)
    end if
   end for
  end if
 for number of rotations Rot_i based on CR_j value **do**
  for var i = 0 to 3 **do**
   $\text{temp}_{f,\,i,\,Rot} \leftarrow \text{Col}_{f,\,i,\,Rot}$
   $\text{Col}_{f,\,i,\,Rot} \leftarrow \text{Col}_{f,\,i,\,Rot}$
   $\text{Col}_{f,\,i,\,Rot} \leftarrow \text{Col}_{f,\,i,\,Rot}$
   $\text{Col}_{f,\,i,\,Rot} \leftarrow \text{Col}_{f,\,i,\,Rot}$
   $\text{Col}_{f,\,i,\,Rot} \leftarrow \text{temp}_{f,\,i,\,Rot}$
  end for
 end for
end for

8) The shiftrows transformation: The ShiftRows operation processes different rows between different faces of the cube. ShiftRows transformation is applied to the matrices that are generated from the ShiftColumns operation. The transformation of shifting the rows of the hybrid cube states over the specified row offsets is denoted as:

ShiftRows(States)

The number of rotations of row vectors in each face with other faces of hybrid cubes depends on last four value of the CR_j. For example, the row vector R_0 of the 4 × 4 matrix of face 1 shifted over the R_0 vector of face 2. The R_0 vector of face 5 is shifted over the R_0 vector of face 1, R_0 vector of face 1 over the face 6, R_0 of face 6 is shifted over the face 3 and the R_0 of face 3 is shifted over the face 5. Similarly, the rows R_1, R_2 and R_3 of each face is shifted based on the coordinate value of the selected rows. The ShiftRows operations can be performed on different faces of the hybrid cube, so the rotation pattern of shifting rows of different faces depends on the CR_j value.

Definition 6. Let the ShiftRows operation be a transposition of row vectors that cyclically shifts the rows of each face over different row offsets of the hybrid cube. If the different faces of a cube are having first and fourth rows then the value of $\delta = 1$, otherwise the middle row's value is $\delta = 0$ depicted in Equation (6). The mathematic formulation of ShiftRows is computed as follows:

$$ShiftRows = \prod_{i=1}^{4} 4Rot_i\left(R_i + \left(\delta \times Q_i\right)\right) \tag{6}$$

where Rot_i is represented as the number of rotations based on the CR_j value, R_i is the row vectors of cube faces and the Q_i is the rotation of quarters Q_1 to Q_4.

Each rotation of Rot_i based on the CR_j value that affects the changes of row vectors rotation into 4 times on different faces of the hybrid cube, because the ShiftRows transformation is applied on four faces of the cube and other two faces rotated clockwise, and counterclockwise based on the ShiftRows transformation. Similarly, if the CR_j value is two then it affects the rows vectors 8 times, and so on. The rotation pattern of rows depends on the last four values (*WXYZ*) of CR_j that rotates the respective rows R_0, R_1, R_2 and R_3 shown in Fig. 6.

For example, the row R_0 is rotated based on the value of *W*. Suppose the value of *W* is 3 then the row vector R_0 has rotated the coordinates into 3 times to the different faces of the cube. Similarly, the value of *X* shifts the coordinates of respective row vector R_1, the value of *Y* shifts the coordinates of row R_2 and the value of *Z* shift the coordinates of respective row R_3. Moreover, if the rotation Rot_i of the cube is having the sided rows (R_0, R_3), then the $\delta = 1$ that affects both sides faces of the cube and it rotates by using the quarter's rotation of Q_i. The rotation pattern of rows of different faces can be divided into two different cases.

a) Case 1: IniFRow value is 1, 3, 5 or 6: If the *IniFRow* value is 1, 3, 5 or 6, then the faces *F1*, *F3*, *F5* and *F6* are utilized as the rotation pattern for the row vectors. The similar process of ShiftRows rotation is performed to the cases of ShiftColumns transformation. Each time the rotation of R_0 of each face affects the face *F2* and it rotates into a counterclockwise. Similarly, the rotation of R_3 in each face affects the face *F4* and it rotates into a clockwise direction.

b) Case 2: IniFRow value is 2 or 4: If the *IniFRow* value is 2 or 4, then the faces *F2*, *F4*, *F5* and *F6* are employed for the row vectors rotation. The rotation of row 1 and row 4 affects the faces *F3* and *F1*, and these faces rotate counterclockwise and clockwise, respectively.

9) Unique matrices operation: In this section, the modulo-16 operation is applied on coordinates of all faces of hybrid cube matrices that are generated from ShiftRows transformation. Each run will give 1 value in the new modulo matrix. The modulo matrices of the hybrid cube which contains the coordinates value are in the range of 0 to 15.

Definition 7. Let the hybrid cube be the 4 × 4 matrices, if any repeated value found in the modulo matrices coordinates, then replace it using the following rules:

$a = a - 1$ for 1st repetition

$a = a - 2$ for 2nd repetition

$a = a - 3$ for 3rd repetition

It will continue until we get zero value. After reaching zero value, if repetition still exists then we will replace by using the following rules:

$a = a + 1$ for 1st repetition

$a = a + 2$ for 2nd repetition

This process will continue until we get the non-repeated matrices value.

Moreover, if the modulo matrices of the hybrid cube are consisting of repeated value(s) in each coordinate of rows and columns, then the properties of Definition 7 are applied on the newly generated modulo matrices in order to get the unique matrices value. These unique matrices will be used to calculate the value of triangular coordinate matrices based on rotation points in the next section.

10) Triangular key matrices: This section calculates the value of key matrices that are generated with ShiftRows transformation using the rotation points which are based on unique matrices. The design of rotation points of *HCs* can be divided into 4 quarters and the rotation points represented as 0 to 15 from quarter Q1 to Q4 [25]. The new key matrices are generated through the calculation of ShiftRows coordinate values based on the rotation points and finally, the value of each matrix is organized based on the unique matrices. The generation of triangular key matrices develops the confusion element in the design of key scheduling, and it increases the difficulty for the cryptanalysis to try all key possibilities. The session keys are generated from master keys by using the TCE quarters rotations that are employed to encrypt the message 1 to 4 in the HiSea encryption algorithm. The novelty of the key schedule algorithm is that all the generated master and session keys of the 3D hybrid cube are invertible and suitable for encryption and decryption in the non-binary block cipher.

B. HiSea Encryption Algorithm

The Hybrid Cube Encryption Algorithm (HiSea) is adopted as the platform in order to validate the proposed key schedule algorithm. The KSAHC algorithm is embedded with the HiSea Encryption algorithm and used to generated encryption keys to encrypt the message into ciphertext. HiSea is the symmetric non-binary block cipher because the encryption and decryption keys, plaintext, ciphertext and internal operation in the encryption or decryption process, are all based on the integer numbers [22], [26]. The Initial Matrix (IM) used during the encryption and decryption process is a secondary security measure which ensures the authenticity of the user. The plaintext is segmented into 64 characters and converted into Extended ASCII codes and the four matrices of Plaintext are represented as *P1* to *P4*. The intermediate result (*P1′*) is obtained from adding *P1* to *P4* with the *IM* and used in the encrypting process of *P2*. The intermediate result (*P2′*) is obtained from adding *P2* with *P1′* and the result is used in the encrypting process of *P3*. This process is repeated for *P4*. The major reason for integrating this method is to ensure any change made in *P1* will reflect in another ciphertext. The process of diffusion is performed using the MixCol and MixRow operations adapted from Toy100 to strengthen the ciphertext [27]. The graphical representation of the encryption process of HiSea block cipher is shown in Fig. 7.

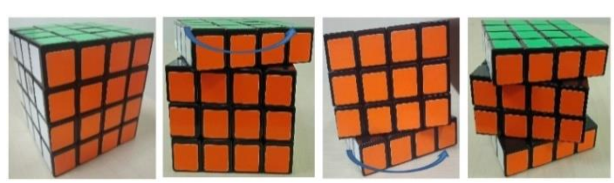

Fig. 6. ShiftRows Transformation.

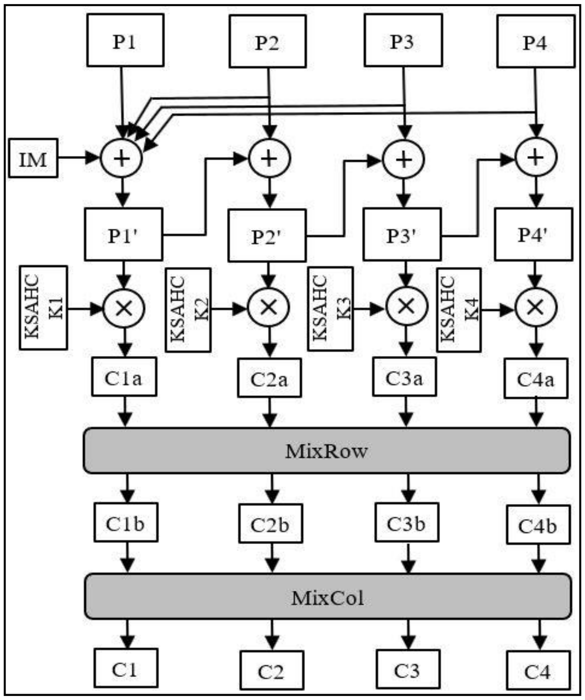

Fig. 7. Encryption Algorithm.

C. Decryption Algorithm

The decryption process is the reverse engineering of the encryption process in which it receives the ciphertext and secret key from the user as the requirement for reconstructing the message back to its readable form [22]. For the purpose of decryption, the recipient required the Ciphertext (C1 to C4) from sender to decrypt the ciphertext into the original readable form. The sender and receiver need to agree earlier on user password for performing the process of encryption and decryption. The password is used to generate the master key in the decryption process, all session keys are the inverse of the encryption keys *K1* to *K4*.

III. Results and Discussion

Let us consider the order 4 hybrid cubes that are used in the rotation of *HCs* which is the main element of the construction of the key scheduling algorithm. In this section, the step by step process of KSAHC algorithm with the example is described and compared with existing algorithms. Furthermore, the generated triangular key matrices have been analyzed to verify the suitability of encryption and decryption keys to the non-binary block cipher. In this regard, some experimental results include the Brute Force, entropy, correlation assessment, avalanche effect, and NIST randomness test suit has been used in the evaluation of the final output of KSAHC algorithm and cipher to prove its strength.

Firstly, the key table is generated from HiSea encryption algorithm in which 24 hybrid cube layers are used as the input to the proposed algorithm. By using the technique [24], 24 matrices are converted into six key matrices that show the six faces of the hybrid cube and these faces were used for the rotation purpose. The generated cube faces of the hybrid cube are shown in Fig. 8.

F2			
7332	5520	3172	6008
7396	5776	3236	5624
3300	5584	7460	5688
3236	5584	7396	5816

F5			
7823	5863	3403	5587
7159	5157	3259	5437
3243	5767	7663	5363
3419	5253	7319	5661

F1			
7871	5995	3451	5583
7151	5093	3251	5485
3227	5835	7647	5295
3475	5253	7375	5773

F6			
5807	5307	5547	5919
5099	5499	5359	4831
5507	5605	5247	4959
5695	5157	5955	6285

F4			
8392	5966	3452	5638
6848	5146	3468	5170
3388	6158	8328	5830
3404	5082	6784	5106

F3			
7849	5529	3169	6161
7101	5803	3461	5191
3393	5881	8073	6257
3237	5451	6877	5095

Fig. 8. Key Matrices Generated using TCE Technique.

Moreover, the string P_j is arbitrary chosen by the user as his private encryption key and is used in the columns and rows selection. Suppose, the value of CR_j is as follows:

P_j = 9876543219

IV = 96421358

m = 123456789

CR_j = (9876543219 + 96421358) mod *123456789*

= 96421457

The value of CR_j shows that the first four numbers are 9642 which are used for the column's rotation and the last four numbers 1457 is used for rows rotations. Furthermore, the value of CR_j is 96421457 that are also used for the initialization face of columns and rows rotation. In this regard, the first four numbers (9642) of CR_j are used to find the initialization face for column rotation. So, Equation (3) is used to calculate the value of *IniFCol* as defined in the following equation:

IniFCol = ((9 + 6 + 4 + 2) mod 6) + 1)= 4

The value of *IniFCol* is 4 which means the initialization face for rotation pattern in ShiftColumns transformation is face 4.

The last four numbers (1457) of CR_j is used for the initialization face of row rotation. In this regard, Equation (4) is used to calculate the value of *IniFRow* described the following equation:

IniFRow = ((1 + 4 + 5 + 7) mod 6) + 1= 6

In this case, the *IniFRow* value is 6, which shows that the row rotation started in ShiftRows operation with face 6.

Based on the *IniFCol* value, the initialization face for the ShiftColumns transformation is face 4, so the rotation pattern can be made using case 1 in ShiftColumns transformation. The CR_j value for the Shifting of columns C_0 to C_3 is 9642. In this case, C_0 vectors rotated 9 times, C_1 vectors rotated 6 times, C_2 vectors rotated 4 times, and C_3 vectors rotated 2 times in the hybrid cube faces 1 to 4. The key matrices are generated using the TCE technique employed for the ShiftColumns transformation. So, case 1 is selected with faces pattern *F1*, *F2*, *F3* and *F4* for the rotation of columns. The rotation of columns C_0 and C_3 affects the other faces *F5* and *F6* that rotated counterclockwise and clockwise respectively based on the TCE quarters rotation technique. The hybrid cube key matrices after the ShiftColumns operation is presented in Fig. 9.

As discussed earlier, the *IniFRow* value for the ShiftRows operation is the face 6, so the rotation pattern can be made based on the ShiftRows case 1. The CR_j value for the Shifting of rows R_0 to R_3 is 1457. In this case, R_0 vectors are rotated 1 time, R_1 vectors rotated 4 times, R_2 vectors rotated 5 times, and R_3 vectors rotated 7 times in the hybrid cube faces 2, 4, 5 and 6. The key matrices generated using the ShiftColumns are used by ShiftRows transformation. So, we selected the faces *F1*, *F3*, *F5* and *F6* for the rotation purpose. Also, the rows R_0 and R_3 affect the other faces *F2* and *F4* that rotated counterclockwise and clockwise respectively. The hybrid cube key matrices after the ShiftRows operation is shown in Fig. 10.

The matrices generated using the ShiftRows transformation are employed for the modulo operation. The modulo matrices of the hybrid cube can be presented in Fig. 11.

The modulo matrices of the hybrid cube that contains the repeated values are depicted in Fig. 10. So, we apply the properties of Definition 7 on the hybrid cube faces in order to remove repetition and generate unique matrices. The resultant unique matrices are shown in Fig. 12. Finally, the output of ShiftRows and unique matrices with the respective coordinate's value are considered in order to calculate the triangular key matrices. The value of triangular key matrices is used according to their coordinates and then calculate the value based on rotation points [25]. The resultant six key matrices of the hybrid cube are used as the encryption and decryption key in the non-binary block cipher. The triangular key matrices are used as the master keys for the encryption process as shown in Fig. 13.

F2			
7871	5966	3172	5638
7151	5146	3236	5170
3227	6158	7460	5830
3475	5082	7396	5106

F5			
5587	5437	5363	5661
3403	3259	7663	7319
5863	5157	5767	5253
7823	7159	3243	3419

F1			
8392	5529	3451	6161
6848	5803	3251	5191
3388	5881	7647	6257
3404	5451	7375	5095

F6			
6285	5955	5157	5695
4959	5247	5605	5507
4831	5359	5499	5099
5919	5547	5307	5807

F4			
7849	5520	3452	6008
7101	5776	3468	5624
3393	5584	8328	5688
3237	5584	6784	5816

F3			
7332	5995	3169	5583
7396	5093	3461	5485
3300	5835	8073	5295
3236	5253	6877	5773

Fig. 9. Hybrid Cube Key Matrices after ShiftColumns Transformation.

F2			
5638	5170	5830	5106
3172	3236	7460	7396
5966	5146	6158	5082
7871	7151	3227	3475

F5			
7332	5995	3169	5583
3403	3259	7663	7319
3300	5835	8073	5295
3404	5451	7375	5095

F1			
5587	5437	5363	5661
6848	5803	3251	5191
5863	5157	5767	5253
5919	5547	5307	5807

F6			
8392	5529	3451	6161
4959	5247	5605	5507
3388	5881	7647	6257
3236	5253	6877	5773

F4			
6008	5624	5688	5816
3452	3468	8328	6784
5520	5776	5584	5584
7849	7101	3393	3237

F3			
6285	5955	5157	5695
7396	5093	3461	5485
4831	5359	5499	5099
7823	7159	3243	3419

Fig. 10. Hybrid Cube Key Matrices after ShiftRows Transformation.

F2			
6	2	6	2
4	4	4	4
14	10	14	10
15	15	11	3

F5			
4	11	1	15
11	11	15	7
4	11	9	15
12	11	15	7

F1			
3	13	3	13
0	11	3	7
7	5	7	5
15	11	11	15

F6			
8	9	11	1
15	15	5	3
12	9	15	1
4	5	13	13

F4			
8	8	8	8
12	12	8	0
0	0	0	0
9	13	1	5

F3			
13	3	5	15
4	5	5	13
15	15	11	11
15	7	11	11

Fig. 11. Modulo-16 Matrices of the Hybrid Cube.

F2			
6	2	5	1
4	3	0	7
14	10	13	9
15	12	11	8

F5			
4	11	1	15
10	9	14	7
3	8	6	13
12	5	2	0

F1			
3	13	2	12
0	11	1	7
6	5	4	8
15	10	9	14

F6			
8	9	11	1
15	14	5	3
12	7	13	0
4	2	10	6

F4			
8	7	6	5
12	11	4	0
1	2	3	9
10	13	14	15

F3			
12	3	5	14
4	2	1	12
14	11	10	9
8	7	6	0

Fig. 12. Hybrid Cube Matrices using unique Matrices Operation.

F2			
8539	6788	4790	9560
2819	2819	2553	3935
11126	6306	8161	9724
2553	1737	1737	3935

F5				F1				F6			
3666	2547	7000	2791	2793	8136	8338	2903	1618	8193	2886	6253
11411	8368	11150	1702	2830	2903	2793	2959	3080	8309	7582	4196
3666	1702	6217	9331	8441	9749	2793	2959	2886	1618	10080	3080
2547	8698	7624	2791	2830	8190	8125	6816	4196	8152	10700	6328

F4			
3924	3924	8408	5186
1618	1618	3004	2908
8596	7358	3004	9989
6185	8376	10948	2908

F3			
7848	3142	9942	2847
3142	8501	6887	1709
7215	1709	5992	9838
3911	3911	7510	2847

Fig. 13. Triangular Key Matrices of the Hybrid Cube.

The novelty of the key schedule algorithm is that all the generated master and session keys of the hybrid cube are invertible and suitable for the encryption and decryption in the non-binary block cipher. After generating the encryption keys using KSAHC algorithm, the encryption of plaintext message with the generated key is performed using the HiSea encryption algorithm [22]. The generated keys have been tested with the existing block cipher (HiSea) and the steps of encryption are demonstrated by using the following message "Hybrid Cubes Encryption Algorithm is originated from UTHM JOHOR." and user password "9876543219". The input message from the user is converted into 64 Extended ASCII codes and the four matrices are represented into 4 × 4 matrices. These matrices are shown as follows:

$$P1 = \begin{bmatrix} 72 & 121 & 98 & 114 \\ 105 & 100 & 32 & 67 \\ 117 & 98 & 101 & 115 \\ 32 & 69 & 110 & 99 \end{bmatrix}, \quad P2 = \begin{bmatrix} 114 & 121 & 112 & 116 \\ 105 & 111 & 110 & 32 \\ 65 & 108 & 103 & 111 \\ 114 & 105 & 116 & 104 \end{bmatrix}$$

$$P3 = \begin{bmatrix} 109 & 32 & 105 & 115 \\ 32 & 111 & 114 & 105 \\ 103 & 105 & 110 & 97 \\ 116 & 101 & 100 & 32 \end{bmatrix}, \quad P4 = \begin{bmatrix} 102 & 114 & 111 & 109 \\ 32 & 85 & 84 & 72 \\ 77 & 32 & 74 & 79 \\ 72 & 79 & 82 & 46 \end{bmatrix}$$

The initial matrix between the sender and receiver is set as follows:

$$IM = \begin{bmatrix} 540 & 3534 & 1872 & 10 \\ 24 & 1710 & 3780 & 442 \\ 3294 & 456 & 10 & 2068 \\ 1886 & 44 & 378 & 3520 \end{bmatrix}$$

The intermediate result (*P1´*) is generated by mixing the *P1*, *P2*, *P3*, *P4* and *IM* as given in Fig. 7. The value of *P1´* is shown as follows:

$$P1' = \begin{bmatrix} 937 & 3922 & 2298 & 464 \\ 298 & 2117 & 4120 & 718 \\ 3656 & 799 & 398 & 2470 \\ 2220 & 398 & 786 & 3801 \end{bmatrix}$$

Session keys are generated using the master key *F2*, the results are as follows:

$$K1 = \begin{bmatrix} 9560 & 3935 & 9724 & 3935 \\ 4790 & 2553 & 8161 & 1737 \\ 6788 & 2819 & 6306 & 1737 \\ 8539 & 2819 & 11126 & 2553 \end{bmatrix} \quad K2 = \begin{bmatrix} 3935 & 1737 & 1737 & 2553 \\ 9724 & 8161 & 6306 & 11126 \\ 3935 & 2553 & 2819 & 2819 \\ 9560 & 4790 & 6788 & 8539 \end{bmatrix}$$

$$K3 = \begin{bmatrix} 2553 & 11126 & 2819 & 8539 \\ 1737 & 6306 & 2819 & 6788 \\ 1737 & 8161 & 2553 & 4790 \\ 3935 & 9724 & 3935 & 9560 \end{bmatrix} \quad K4 = \begin{bmatrix} 8539 & 6788 & 4790 & 9560 \\ 2819 & 2819 & 2553 & 3935 \\ 11126 & 6306 & 8161 & 9724 \\ 2553 & 1737 & 1737 & 3935 \end{bmatrix}$$

The intermediate result of *P1´* is then mixed with *K1*. The matrix *C1a* is given as follows:

$$C1a = \begin{bmatrix} 47305020 & 21486039 & 60772482 & 15675827 \\ 47086872 & 20215653 & 54143777 & 13839353 \\ 62571524 & 24511099 & 72062591 & 22771459 \\ 60921727 & 22682547 & 72081800 & 20496261 \end{bmatrix}$$

After that, MixRow function is applied on *C1a*, the matrix *C1b* is mentioned as follows:

$$C1b = \begin{bmatrix} 84466886 & 129563541 & 97934348 & 123753329 \\ 81141878 & 121446302 & 88198783 & 115070002 \\ 109854082 & 159145214 & 119345149 & 157405574 \\ 104100535 & 155686074 & 115260608 & 153499788 \end{bmatrix}$$

Furthermore, the MixCol function is applied on *C1b* and generates Ciphertext *C1* from Plaintext message P1 is given as follows:

$$C1 = \begin{bmatrix} 269709299 & 406695917 & 301393739 & 392323119 \\ 275462846 & 410155057 & 305478280 & 396228905 \\ 295096495 & 436277590 & 322804540 & 425975364 \\ 298421503 & 444394829 & 332540105 & 434658691 \end{bmatrix}$$

Similarly, the Plaintext message *P2*, *P3* and *P4* follows the similar process as for *P1* and generate the final ciphertext *C2*, *C3* and *C4*. The Ciphertext (*C2* to *C4*) are written as follows:

$$C2 = \begin{bmatrix} 417694123 & 372785950 & 363286275 & 417594055 \\ 406785499 & 361116578 & 352065797 & 403183993 \\ 376710203 & 341235079 & 323736215 & 385917619 \\ 413865098 & 369410576 & 355765496 & 418881909 \end{bmatrix}$$

$$C3 = \begin{bmatrix} 436820789 & 337379515 & 451904811 & 302898477 \\ 438423640 & 337460276 & 453995546 & 301145893 \\ 468659513 & 359242079 & 479584329 & 322486079 \\ 479934787 & 364471642 & 491438341 & 332868618 \end{bmatrix}$$

$$C4 = \begin{bmatrix} 443632423 & 387993015 & 396962894 & 448294179 \\ 460488355 & 401090600 & 410736380 & 461558698 \\ 442555760 & 377415801 & 390990607 & 439180809 \\ 429290483 & 361006024 & 379718683 & 420514922 \end{bmatrix}$$

Based on the obtained results, the key schedule algorithm and the ciphertext *C1* to *C4* have been evaluated and tested. The results are described in the following sub-sections.

A. Brute Force Attack

In general, this attack is possible if an adversary could generate one possible correct key from a large key space. The attacker has no knowledge of encryption key(s), so the attacker generates and computes every possible combination of the encryption key to recover the secret key that was used for the encryption process. In order to achieve the optimum security level, the key space must be at least 2^{128} to resist the Brute Force attack [28]. In this cipher, the KSAHC encryption keys are represented as $n \times n \times n$ matrix of integer numbers and used in the development of the permutation and substitution of order 4 square matrix. Each entry of encryption key lies between the range of from 1618 to 11198 or within 2^{14} bits (approx.). So, the key space for encryption and decryption keys calculated as follows:

$$2^{14} \times 2^{14} \times 2^{14} \times \ldots\ldots 2^{14} = (2^{14})^{16} = 2^{224}$$

or approximately the number of alternative keys

$= 2.70 \times 10^{67}$ keys

Furthermore, the comparison of AES, DKSA, HiSea and the proposed algorithm based on the Brute Force has been calculated and presented in Table II. The number of alternative keys shows that the KSAHC algorithm is computationally secured and has a large key space which makes the brute-force attack difficult and time-consuming. Hence, the number of keys used in the HiSea with KSAHC algorithm can determine the practically infeasible to conduct a brute-force attack due to the limitation of computational power and length of the time.

B. Entropy

In this test, the strength of overall implementation of KSAHC is estimated by using random matrix technique. The strength of the master key matrices of proposed KSAHC algorithm is calculated by using MATLAB function CalculateEnt() and compared with the HiSea Key Schedule Algorithm (KSA). The KSAHC triangular key matrices shown in Fig. 13 are used to estimate the normalized Shannon entropy.

The average normalized Shannon entropy for the HiSea KSA matrices are 0.8491 and the proposed KSAHC triangular matrices are 0.9466 as all the entropy results should be closer to 1 rather than 0. The result shows that the strength of proposed KSAHC triangular key matrices is better than the HiSea KSA as shown in Table III. Hence, each matrix of triangular key matrices that represents the hybrid cube blocks consist of 16 decimal numbers, is average of 94.66% random which can be considered as almost random and it is suitable for the development of non-binary block cipher.

The result obtained from the entropy test has been compared with its AES, HiSea and DKSA counterparts. For the purpose of comparison between these block cipher, four different ciphertexts from slightly different input keys were generated and the encryption process performed with a similar message with each of the generated keys.

Based on the results from Fig. 14, the average entropy of AES has 0.9273, HiSea has 0.9830, DKSA has 0.9367, while the proposed cipher has higher average entropy of 0.9968. The results show that the generated ciphertext produces highly random output that makes it difficult for the cryptanalyst to observe the behavior and changes on the output for the purpose of attack as all the outputs are different and did not reveal any relationship between one another.

C. Correlation Assessment

The correlation test has been conducted between the blocks of encryption keys (*F1* to *F6*) and their session keys generated using KSAHC algorithm, to figure out if any predictable pattern exists among them. The value of predictable pattern between the encryption keys and session keys should be closer to zero rather than 1, because if the result is closer to 1 then it is easy for a cryptanalyst to predict other keys due to more similarities. In this test, two sets of keys are required for the testing purpose and all 4 session keys are employed to determine whether the similarity exists among the keys or not. The four-session keys of all faces (*F1* to *F6*) of the hybrid cube can be represented as *S_K1*, *S_K2*, *S_K3* and *S_K4* shown in Table IV. The average correlation between all session keys of different faces of the 3D hybrid cube appears as -0.009472 which is closer to 0 and that means there is no correlation exist between the session keys, thus makes the related key attack very difficult.

TABLE. II. COMPARISON OF KEY SPACES BASED ON THE BRUTE FORCE

Key size (bits)	**Algorithm**	**No. of Alternative Keys**	**Time required at 10^9 decryption (years)**	**Time required at 10^{13} decryption (years)**
128	AES	$2^{128} = 3.4 \times 10^{38}$	2^{127} ns $= 5.3 \times 10^{21}$	5.3×10^{17}
128	DKSA	$2^{128} = 3.4 \times 10^{38}$	2^{127} ns $= 5.3 \times 10^{21}$	5.3×10^{17}
192	HiSea	$2^{192} = 6.3 \times 10^{57}$	2^{191} ns $= 9.8 \times 10^{40}$	9.8×10^{36}
224	KSAHC	$2^{224} = 2.70 \times 10^{67}$	2^{223} ns $= 8.6 \times 10^{49}$	8.6×10^{47}

TABLE. III. NORMALIZE SHANNON ENTROPY OF PROPOSED KSAHC ALGORITHM

Keys	**HiSea KSA**	**Proposed KSAHC**
K1	0.8448	0.9514
K2	0.8644	0.9426
K3	0.8430	0.9520
K4	0.8476	0.9463
K5	0.8473	0.9385
K6	0.8473	0.9492
Average	0.8491	0.9466

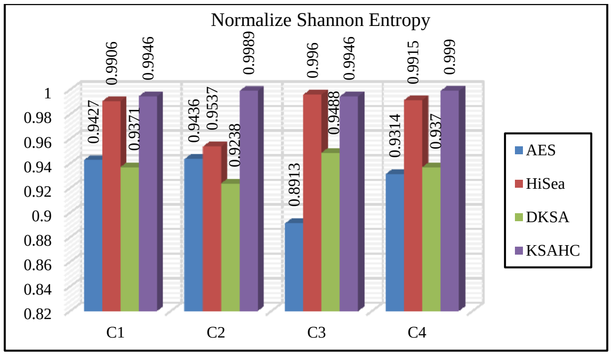

Fig. 14. Entropy Test with the different Encryption Algorithm.

In Table V, the proposed KSAHC algorithm has an average correlation assessment of -0.000601 while it ultimately compared with the AES, HiSea and DKSA, they have the average correlation of 0.185622, -0.0779 and -0.0419, respectively. The proposed KSAHC has outperformed the most widely used AES algorithm in terms of correlation. It is concluded that all the session keys are individually generated and there exist no relationship with each other. It is difficult for cryptanalyst to conduct a related key attack, even if the cryptanalyst manages to get one session key, but the other keys are unrelated and independent, so it is not easy to get all keys. The graphical representation of the correlation test between the different algorithms is shown in Fig. 15.

TABLE. IV. COMPARISON OF SESSION KEYS OF DIFFERENT FACES OF THE HYBRID CUBE

Faces	x	y	Correlation		Faces	x	y	Correlation
F1	S_K1_F1	S_K2_F1	-0.191778		F4	S_K1_F4	S_K2_F4	0.146547
	S_K1_F1	S_K3_F1	-0.362025			S_K1_F4	S_K3_F4	-0.103435
	S_K1_F1	S_K4_F1	0.535537			S_K1_F4	S_K4_F4	0.147765
	S_K2_F1	S_K3_F1	0.111523			S_K2_F4	S_K3_F4	0.147007
	S_K2_F1	S_K4_F1	-0.117235			S_K2_F4	S_K4_F4	-0.138630
	S_K3_F1	S_K4_F1	-0.0082772			S_K3_F4	S_K4_F4	-0.148100
F2	S_K1_F2	S_K2_F2	-0.022596		F5	S_K1_F5	S_K2_F5	0.140618
	S_K1_F2	S_K3_F2	0.127745			S_K1_F5	S_K3_F5	-0.055084
	S_K1_F2	S_K4_F2	-0.092219			S_K1_F5	S_K4_F5	-0.149065
	S_K2_F2	S_K3_F2	-0.022596			S_K2_F5	S_K3_F5	-0.117827
	S_K2_F2	S_K4_F2	0.070664			S_K2_F5	S_K4_F5	0.035217
	S_K3_F2	S_K4_F2	-0.076288			S_K3_F5	S_K4_F5	0.296118
F3	S_K1_F3	S_K2_F3	-0.081496		F6	S_K1_F6	S_K2_F6	-0.356507
	S_K1_F3	S_K3_F3	-0.272481			S_K1_F6	S_K3_F6	0.221452
	S_K1_F3	S_K4_F3	-0.081801			S_K1_F6	S_K4_F6	-0.326664
	S_K2_F3	S_K3_F3	-0.08109			S_K2_F6	S_K3_F6	-0.356356
	S_K2_F3	S_K4_F3	0.272784			S_K2_F6	S_K4_F6	0.193601
	S_K3_F3	S_K4_F3	-0.081034			S_K3_F6	S_K4_F6	0.455002
The average correlation between the face F1 to F6								-0.009472

TABLE. V. COMPARISONS OF DIFFERENT ALGORITHMS BASED ON CORRELATION

x	y	AES	HiSea	DKSA	Proposed KSAHC
S_K1	S_K2	0.501405	0.52327	0.041818	0.359631
S_K1	S_K3	0.389243	-0.65528	-0.42882	0.207383
S_K1	S_K4	0.353688	-0.85956	0.123036	-0.361394
S_K2	S_K3	-0.22499	0.979662	0.076945	0.008391
S_K2	S_K4	0.05817	0.460786	0.173298	-0.063184
S_K3	S_K4	0.036223	-0.91629	-0.23768	-0.154434
Average Correlation		0.185622	-0.0779	-0.0419	-0.000601

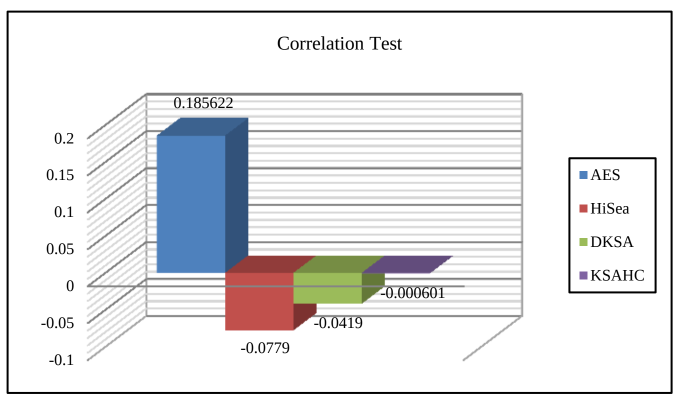

Fig. 15. Comparative Analysis of Average Correlation of different Algorithms.

D. *Avalanche Effect*

This test describes the behavior of the algorithm which determines the slight changes in the input that significantly affects the output value. In other words, the avalanche effect is used to measure the dissimilarity between the input and output changes. If block cipher exhibits ineffective avalanche property, the output would not be random and independently generated and the cryptanalyst could easily exploit and predict the input from the output. An efficient avalanche property of a cryptographic algorithm should be greater than or equal to 50% (≥ 50%) [29].

In this test, several input strings of our proposed algorithm have been tested to verify how much it affects the output by changing the beginning, middle and end of the inputs as shown in Table VI. The avalanche test of the proposed algorithm with different inputs produced an average result of 93% that means the proposed algorithm has favorable avalanche effect compared to others.

The avalanche effect of the session keys of different faces of the hybrid cube was tested and the proposed algorithm produces entirely different session keys in each rotation. As we mentioned earlier, two set of session keys used to calculate the avalanche effect of face *F1* to *F6* to examine the difference in the output. Also, the four-session keys from each face were compared with their counterpart session keys of different inputs, all session keys appear to be non-linear and different from each other. Hence, the average correlation of face *F1* to *F6* shown in Table VII proves that the proposed algorithm has stronger avalanche property.

A comparison of four cryptographic algorithms (AES, HiSea, DKSA and proposed KSAHC) based on different set of round keys or session keys were generated and tested to observe the changes in each session key. Based on the test, DKSA appears as the lower avalanche test score of 88% while the AES and HiSea having the avalanche score of 90%. On the other hand, the proposed KSAHC achieved the highest score in avalanche test that is 93% which means that the proposed KSAHC algorithm always produces a different set of keys for every small change in the input.

TABLE. VI. AVALANCHE EFFECT OF PROPOSED ALGORITHM WITH DIFFERENT INPUTS

Inputs	**Output Keys**	**Aval.**
0876543219	2793 8136 7066 1709 1709 6126 2830 3911 8441 5032 2793 9194 11150 2830 9737 3911	94 %
8876543219	8200 1618 4794 8056 2819 9697 3924 1618 2553 9679 3924 8075 2819 2553 7959 7249	
9876543219	1618 8193 2886 6253 3080 8309 7582 4196 2886 1618 10080 3080 4196 8152 10700 6328	91 %
9886543219	3666 3142 11150 2959 3666 7959 1618 6460 8056 10145 8790 3142 5032 2959 1618 6102	
9876543219	3004 2908 11206 1737 9194 8136 10002 8896 1737 6126 6900 3935 5181 3935 2908 3004	95%
9876653219	2886 8170 6788 1618 2791 9974 2791 8063 1618 6913 6310 2886 9647 3924 8541 3924	
9876653219	1618 2553 2553 1618 11560 4196 3004 11198 8055 5076 6190 4196 6932 7275 3004 9852	94%
9876653210	3935 1737 1737 3935 6144 4196 3004 11198 9697 5076 8541 4196 8170 6932 7275 3004	
1234567890	2959 8487 2903 5181 4196 9942 4196 10080 2903 2959 3080 3080 7215 9130 6328 8075	94%
1234567891	7848 3142 7582 2847 8309 2903 3142 8578 2903 8056 8226 2959 2959 7510 7959 2847	
2234567891	3912 1710 7067 2831 2794 10077 7849 3912 2794 8187 7511 9119 1710 8043 6306 2831	89%
3234567891	2887 2887 2792 4196 1702 6306 11199 8171 4196 5077 8542 2792 7111 9395 9119 1702	
Average Avalanche		93%

TABLE. VII. AVALANCHE EFFECT OF DIFFERENT FACES OF THE HYBRID CUBE

Faces	**Key and Session Key**		**Avalanche**		**Faces**	**Key and Session Key**		**Avalanche**
F1	S_K1_F1	S_K2_F1	94%		F4	S_K1_F4	S_K2_F4	92%
	S_K1_F1	S_K3_F1	94%			S_K1_F4	S_K3_F4	95%
	S_K1_F1	S_K4_F1	94%			S_K1_F4	S_K4_F4	92%
	S_K2_F1	S_K3_F1	93%			S_K2_F4	S_K3_F4	94%
	S_K2_F1	S_K4_F1	97%			S_K2_F4	S_K4_F4	89%
	S_K3_F1	S_K4_F	94%			S_K3_F4	S_K4_F4	89%
F2	S_K1_F2	S_K2_F2	91%		F5	S_K1_F5	S_K2_F5	91%
	S_K1_F2	S_K3_F2	85%			S_K1_F5	S_K3_F5	97%
	S_K1_F2	S_K4_F2	94%			S_K1_F5	S_K4_F5	92%
	S_K2_F2	S_K3_F2	94%			S_K2_F5	S_K3_F5	94%
	S_K2_F2	S_K4_F2	86%			S_K2_F5	S_K4_F5	94%
	S_K3_F2	S_K4_F2	92%			S_K3_F5	S_K4_F5	91%
F3	S_K1_F3	S_K2_F3	97%		F6	S_K1_F6	S_K2_F6	95%
	S_K1_F3	S_K3_F3	91%			S_K1_F6	S_K3_F6	92%
	S_K1_F3	S_K4_F3	92%			S_K1_F6	S_K4_F6	97%
	S_K2_F3	S_K3_F3	91%			S_K2_F6	S_K3_F6	98%
	S_K2_F3	S_K4_F3	95%			S_K2_F6	S_K4_F6	94%
	S_K3_F3	S_K4_F3	97%			S_K3_F6	S_K4_F6	95%
Average avalanche test between the face F1 to F6								93%

Moreover, the promising results show that the related key attack and ciphertext-only attack will be extremely difficult or even impossible. The KSAHC algorithm shows a very good avalanche property as compared with the existing algorithms as shown in Fig. 16.

E. The NIST Test

In order to analyze the randomness of the proposed scheme, the statistical test suite developed by the NIST is used. The purpose of the NIST test suite is to determine the randomness of a sequence. Any cryptographic algorithm is considered to pass the NIST test, if the resulting P-value is greater than 0.01 then it is said to be the random [30]. For that purpose, three statistical tests (frequency mono-bit test, block frequency test, and runs test) have been conducted on the output of the proposed scheme. Based on Table VIII, the results show that the P-value for the outputs of the keys of six faces of the Hybrid cube is greater than 0.01. Hence, it can be concluded that the results are in favor of the proposed algorithm and the sequence generated by the proposed KSAHC algorithm are random.

Similarly, the comparison of AES, HiSea, DKSA and proposed KSAHC has been conducted based on the NIST test to figure out the randomness of the proposed scheme as compared to other cryptographic algorithms. Table IX shows that the comparison of proposed KSAHC with different cryptographic algorithms based on the frequency test has outperformed AES, HiSea, and DKSA in term of randomness. Meanwhile, Table X shows that the proposed algorithm has achieved a better result as compared to AES and DKSA but the HiSea appears to have outperformed the proposed algorithm in terms of block frequency test. Also, Table XI shows that the comparison based on the NIST runs test in which the proposed algorithm performed better compared to HiSea but the DKSA and AES appears to have better results compared to the proposed algorithm.

The proposed KSAHC algorithm shows a better result in frequency and block frequency test as compared to AES and DKSA but the HiSea shows the better result in the block frequency test. While the results of the proposed scheme underperformed in the Runs test.

Fig. 17 shows that the average results of the frequency test are 0.6935, 0.5921 in frequency block test and 0.6486 in the run test. So, the requirement of the pseudorandom number generator has been achieved by the proposed algorithm. Hence, the results of diffusion test are in favor of the proposed algorithm and the generated sequence has an efficient diffusion property.

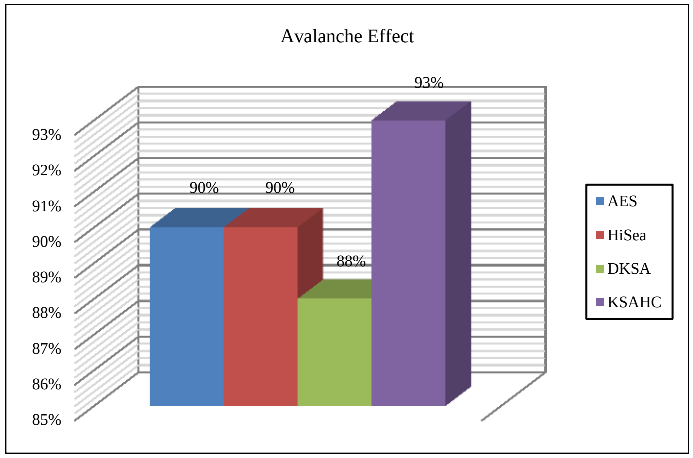

Fig. 16. The Average Result of Avalanche Test on Four different Algorithms.

TABLE. VIII. NIST TEST ANALYSIS OF PROPOSED ALGORITHM

Faces Key ID	**Frequency Test**	**Block Frequency Test**	**Runs Test**		**Faces Key ID**	**Frequency Test**	**Block Frequency Test**	**Runs Test**
S_K1_F1	0.2485	0.5745	0.1791		S_K1_F4	0.8918	0.9181	0.7845
S_K2_F1	0.1559	0.1266	0.7324		S_K2_F4	0.2185	0.1621	0.9733
S_K3_F1	0.4142	0.1660	0.0621		S_K3_F4	0.1950	0.3585	0.4629
S_K4_F1	0.5862	0.6850	0.5719		S_K4_F4	0.0656	0.7673	0.8025
S_K1_F2	0.6299	0.5474	0.8237		S_K1_F5	0.7331	0.4631	0.8316
S_K2_F2	0.2164	0.1350	0.8642		S_K2_F5	0.2463	0.6329	0.3969
S_K3_F2	0.1483	0.0748	0.0316		S_K3_F5	0.5862	0.7863	0.8008
S_K4_F2	0.7319	0.3585	0.4460		S_K4_F5	0.9454	0.1229	0.3042
S_K1_F3	0.6817	0.6045	0.9908		S_K1_F6	0.6817	0.4631	0.9908
S_K2_F3	0.4962	0.7623	0.0129		S_K2_F6	0.8379	0.6887	0.2474
S_K3_F3	0.2463	0.2283	0.8019		S_K3_F6	0.4163	0.6121	0.7210
S_K4_F3	0.9456	0.1229	0.8376		S_K4_F6	0.8379	0.6045	0.6350
Mean	0.4584	0.3655	0.5295		Mean	0.5546	0.5483	0.6626
Average NIST test between the face F1 to F6						0.5065	0.4569	0.5960

TABLE. IX. COMPARISON OF THE DIFFERENT ALGORITHMS BASED ON NIST FREQUENCY TEST

Key ID	AES	HiSea	DKSA	Proposed KSAHC
K1_1	0.3768	0.7022	0.6171	0.6817
K1_2	0.0205	0.5924	0.6171	0.8379
K1_3	1.0000	0.4913	0.4386	0.4163
K1_4	0.5959	0.9390	1.0000	0.8379
Average	0.4983	0.6812	0.6682	0.6935

TABLE. X. COMPARISON OF THE DIFFERENT ALGORITHMS BASED ON BLOCK FREQUENCY TEST

Key ID	AES	HiSea	DKSA	Proposed KSAHC
K1_1	0.7776	0.5213	0.1512	0.4631
K1_2	0.3189	0.9396	0.2017	0.6887
K1_3	0.6728	0.7409	0.2906	0.6121
K1_4	0.4884	0.4898	0.4335	0.6045
Average	0.5644	0.6729	0.2693	0.5921

TABLE. XI. COMPARISON OF THE DIFFERENT ALGORITHMS BASED ON NIST RUNS TEST

Key ID	AES	HiSea	DKSA	Proposed KSAHC
K1_1	0.4123	0.9479	0.9750	0.9908
K1_2	0.4931	0.9564	0.9750	0.2474
K1_3	1.0000	0.4675	0.2621	0.7210
K1_4	0.8790	0.1461	0.6171	0.6350
Average	0.6961	0.6295	0.7073	0.6486

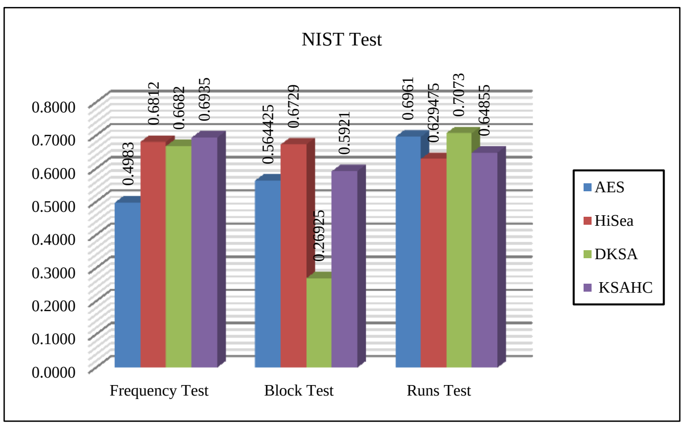

Fig. 17. Comparison of the different Algorithms based on NIST Test.

IV. CONCLUSION AND FUTURE WORK

In this paper, the KSAHC algorithm based on TCE technique is presented that is used to generate the encryption and decryption keys for the non-binary block cipher. The permutation and combination of the 3D hybrid cube from the set of integers, triangular coordinate extraction technique, and rotation of HCs are used in the design of KSAHC algorithm and it is suitable for HiSea encryption algorithm. The Brute Force and entropy test were carried out to demonstrate the strength of the keys which is highly random and having the large key space that makes it difficult and time-consuming for predicting any key pattern. The average result of the correlation is -0.000601 that closer to zero which shows no correlation exists between the input and output. Also, the result of avalanche effect is 93% which means that the proposed algorithm always produces a different set of keys for every small change in the input and it makes the attack extremely difficult or even impossible on the proposed KSAHC algorithm. Furthermore, the NIST test used to analyze the randomness of the sequences generated by the proposed scheme. The frequency mono-bit test, block frequency test and runs test has been conducted and the result of *P*-value obtained is greater than 0.01. Hence, it can be concluded that the results from the diffusion test are in favor of the proposed algorithm and the sequence generated by the proposed KSAHC algorithm has an efficient diffusion property. The results obtained from this analysis are employed to improve the overall design of the HiSea encryption algorithm. In future work, the proposed algorithm for non-binary block cipher produces only 128 bits keys but it can be upgraded into 256 and 512-bit keys that will enhance the security and performance of the algorithm. Furthermore, the proposed algorithm will be upgraded into Authenticated Encryption because of its outstanding performance.

ACKNOWLEDGMENT

The authors would like to thank the Universiti Tun Hussein Onn Malaysia and Ministry of Higher Education (MOHE) Malaysia for supporting this research under Fundamental Research Grant Scheme (FRGS), Vote No. 1642.

REFERENCES

[1] M. Ebrahim, S. Khan, and U. Bin Khalid, “Symmetric Algorithm Survey: A Comparative Analysis,” Int. J. Comput. Appl., vol. 61, no. 20, pp. 12–19, 2013.

[2] L. Savu, “Cryptography Role in Information Security,” Recent Res. Commun. Inf. Technol., pp. 36–41, 2011.

[3] A. H. Disina, Z. A. Pindar, and S. Jamel, “Enhanced Caeser Cipher to Exclude Repetition and Withstand Frequency Cryptanalysis,” J. Netw. Inf. Secur., 2015.

[4] J. Daemen and V. Rijmen, The Design of Rijndael - The Advanced Encryption Standard. 2002.

[5] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, and M. M. Deris, “A Comprehensive Survey on the Cryptographic Encryption Algorithms,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 11, pp. 333–344, 2017.

[6] M. F. Mushtaq, U. Akram, I. Khan, S. N. Khan, A. Shahzad, and A. Ullah, “Cloud Computing Environment and Security Challenges: A Review,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 10, pp. 183–195, 2017.

[7] S. Jamel, T. Herawan, and M. M. Deris, “A cryptographic algorithm based on hybrid cubes,” Comput. Sci. Its Appl. ICCSA, vol. 6019, pp. 175–187, 2010.

[8] A. H. Disina, S. Jamel, M. Aamir, Z. A. Pindar, M. M. Deris, and K. M. Mohamad, “A Key Scheduling Algorithm Based on Dynamic Quasigroup String Transformation and All-Or- Nothing Key Derivation Function,” J. Telecommun. Electron. Comput. Eng., vol. 9, no. 3–5, pp. 1–6, 2017.

[9] N. I. of S. NIST, Advanced Encryption Standard (AES). 2001.

[10] Q.-A. Kester, “A Hybrid Cryptosystem Based on Vigenere Cipher and Columnar Transposition Cipher,” Int. J. Adv. Technol. Eng. Res., vol. 3, no. 1, pp. 141–147, 2013.

[11] R. Hoda, “Finding the total number of legal permutations of the Rubik ’ s Cube,” in Extended Essay–Mathematics, Trondheim Katedralskole, 2010, pp. 1–32.

[12] G. Hanchinamani and L. Kulakarni, “A new approach for image encryption based on cyclic rotations and multiple blockwise diffusions using Pomeau-manneville and sin maps,” J. Comput. Sci. Eng., vol. 8, no. 4, pp. 187–198, 2014.

[13] B. Nini and D. Bouteldja, “Virtual Cylindrical View of a Color Image for its Permutation for an Encryption Purpose,” Int. J. Comput. Appl., vol. 16, no. 1, pp. 11–17, 2011.

[14] J. Shen, X. Jin, and C. Zhou, “A color image encryption algorithm based on magic cube transformation and modular arithmetic operation,” Adv. Multimed. Inf. Process., vol. 3768, pp. 270–280, 2005.

[15] N. Anghel, “Determinant Identities and the Geometry of Lines and Circles,” Analele Stiint. Ovidius Constanta, Versita, vol. 22, no. 2, pp. 37–49, 2014.

[16] D. Hearn and M. P. Baker, Computer Graphics - C version. Pearson Education, 2005.

[17] W. Kuhnel, Differential Geometry, Third Edit. American Mathematical Society, 2015.

[18] M. Trenkler, “An algorithm for making magic cubes,” Pi ME J., vol. 12, no. 2, pp. 105–106, 2005.

[19] A. V. Diaconu and K. Loukhaoukha, “An improved secure image encryption algorithm based on rubik’s cube principle and digital chaotic cipher,” Math. Probl. Eng., pp. 1–10, 2013.

[20] A. B. Abugharsa, A. S. B. H. Basari, and H. M. Almangush, “A New Image Scrambling Technique using Block Rotation Algorithm based on Rubik ’ s Cube,” Aust. J. Basic Appl. Sci., vol. 7, no. 14, pp. 97–108, 2014.

[21] D. Rajavel and S. Shantharajah, “Scrambling algorithm for encryption of text using cube rotation artificial intelligence technique,” Biomed. Res., pp. 251–256, 2016.

[22] S. Jamel, M. M. Deris, I. T. R. Yanto, and T. Herawan, “The hybrid cubes encryption algorithm (HiSea),” Commun. Comput. Inf. Sci. Springer-Verlag Berlin Heidelb., vol. 154, pp. 191–200, 2011.

[23] D. Rajavel and S. P. Shantharajah, “Cryptography Based on Combination of Hybridization and Cube ’ s Rotation,” Int. J. Comput. Intell. Informatics, vol. 1, no. 4, pp. 294–299, 2012.

[24] M. F. Mushtaq, S. Jamel, and M. M. Deris, “Triangular Coordinate Extraction (TCE) for Hybrid Cubes,” J. Eng. Appl. Sci., vol. 12, no. 8, pp. 2164–2169, 2017.

[25] M. F. Mushtaq, S. Jamel, K. M. Mohamad, S. K. A. Khalid, and M. M. Deris, “Key Generation Technique based on Triangular Coordinate Extraction for Hybrid Cubes,” J. Telecommun. Electron. Comput. Eng., vol. 9, no. 3–4, pp. 195–200, 2017.

[26] S. Jamel, M. M. Deris, I. Tri, R. Yanto, and T. Herawan, “HiSea : A Non Binary Toy Cipher,” J. Comput., vol. 3, no. 6, pp. 20–27, 2011.

[27] L. Granboulan, E. Levieil, and G. Piret, “Pseudorandom Permutation Families over Abelian Groups,” Fast Softw. Encryption, vol. 4047, pp. 57–77, 2006.

[28] A. Akhavan, A. Samsudin, and A. Akhshani, “A novel parallel hash function based on 3D chaotic map,” EURASIP J. Adv. Signal Process., vol. 2013, no. 1, pp. 1–12, 2013.

[29] J. Ahmad and F. Ahmed, “Efficiency analysis and security evaluation of image encryption schemes,” Int. J. Video Image Process. Netw. Secur., vol. 12, no. 4, pp. 18–31, 2012.

[30] A. Rukhin et al., “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” Natl. Inst. Stand. Technol., pp. 1–82, 2010.