Data Security: A New Symmetric Cryptosystem based on Graph Theory

Khalid Bekkaoui, Soumia Ziti, Fouzia Omary
Intelligent Processing and Security of Systems(IPSS)
Faculty of Sciences, Mohammed V University
in Rabat, Morocco

***Abstract*—Sharing private data in an unsecured channel is extremely critical, as unauthorized entities can intercept it and could break its privacy. The design of a cryptosystem that fulfills the security requirements in terms of confidentiality, integrity and authenticity of transmitted data has therefore become an unavoidable imperative. Indeed, a lot of work has been carried out in this regard. Although many cryptosystems have been proposed in the published literature, it has been found that their robustness and performance vary relatively from one to another. Adopting this reflection, we address in this paper the concept of block cipher, which is a major cryptographic solution to guarantee confidentiality, by involving the properties of graph theory to represent the plaintext message. Our proposal is in fact a new symmetric encryption block cipher that proceeds by representing plaintext messages using disjoint Hamiltonian circuits and then dealing with them as an adjacency matrix in a pre-encryption phase. The proposed system relies on a particular sub-key generator that has been carefully designed to produce the encryption keys according to the specifications of the system. The obtained experimental results demonstrate that our proposed cryptosystem is robust against statistical attacks, particularly the DIEHARD test, and presents both good confusion and good diffusion.**

***Keywords*—Cryptosystem; graph theory; hamiltonian circuits; adjacency matrix; block cipher; encryption**

I. Introduction

Cryptography is a component of cryptology that is based on a number of methods and principles for converting a readable message to a totally unreadable one. This field is dealing with many security problems such as the confidentiality of communications via non-secure channels, the privacy of individuals, the data storing on unsecured mediums, and so on. Cryptography refers to the study and analysis of data encryption systems intended to reduce the impact of hackers and to prevent, as best as possible, any unauthorized attempts to gain access to these confidential data. The main principles of information security, notably confidentiality, integrity, authentication, and non-repudiation [1].

Confidentiality is a crucial part of security. It can be ensured by an encryption process, whereby the data becomes non-intelligible to any non-authorized parties trying to gain access to it. The idea behind of encryption process is to transform a plaintext into a ciphertext, so only authorized parties can obtain the message in its original format by reversing the encryption process, known as decryption. Technically, decryption should be extremely difficult for any unauthorized and unqualified parties attempting to perform it.

Over the years, cryptography has continued to be improved and has progressively became an indispensable part for private data sharing. All contributions dedicated to this field of research have aroused great interest. In the literature, cryptography can be classified into three categories: Symmetric Key Cryptography, which is an encryption system where both the transmitter and the recipient of the message use one common key such as DES [2], AES [3], or IDEA [4], to encrypt and decrypt the messages. The second category is Asymmetric Key Cryptography. In this system, a couple of keys(private and public keys) are used in order to encrypt and decrypt the messages such as RSA [5], ElGamal [6], Diffie-Hellman [7], etc. The last category is Hybrid key Crypthography, Which consist of using an encryption mode that utilizes both symmetric and asymmetric public key encryption. This method benefits from public key cryptography for key sharing and from the speed of symmetric encryption for message encryption. Nowadays, cryptology is able to handle a substantial set of mathematical tools, that allowed for improvements in terms of efficiency and performance. In particular, graph theory is a field that is considered very promising in this regard, since it provides concepts that could be useful in solving problems in every network related areas.

Graph theory in mathematics refers to the study of graphs, which are a major object of discrete mathematics. Generally, a graph is represented as a set of vertices linked by edges. They are thus mathematical structures used for modelling pair-wise relationships between objects. It can be found in road networks, electrical circuits, constellations, etc. Graphs provide a way of thinking that can be used for modeling a vast range of problems. They are the foundation of numerous computer programs that allow communication and advanced technological processes. The seven bridges of Konigsberg (1736) [8] is a mathematical problem well known for having established the foundations of the theory of graphs. Graph theory is a relatively new concept that has been successfully incorporated and has enabled the development of stronger encryption algorithms that have proven to be difficult to break, even for the latest software solutions. In fact, it consists of modeling encryption problems by graph representation, so that they eventually become problems in graph theory where the solutions are usually well-known. Although solutions to graph problems can be fairly easy and efficient (with respect to the time required for computational processing which is reasonable), they can also be rather difficult (relative to the processing time increases exponentially). This resulted in the application of concepts introduced in graph theory to large-scale cryptography, since many NP-hard problems are derived

from this theory.

Considering the above mentioned points, the design of cryptosystems based on the concepts of graph theory is of utmost importance. In this work, we present a new cryptosystem that takes advantage of the principles of graph theory, which enable a high degree of security while maintaining the performance of data processing. The main idea of our approach is to represent the plaintext with all disjoint Hamiltonian circuits as a pre-encryption phase, then using our own sub-keys generator following the cipher block chaining mode of operation to encrypt the plaintext.

The rest of the paper is structured as follows. Section 2 presents preliminary knowledge. Section 3 presents a literature review of related work. Section 4 details the proposed scheme. Security analysis and experimental results are elaborated in Section 5, and lastly, the conclusion and future works is given in Section 6.

II. Preliminary Knowledge

- **Graph**: A graph G is a set of points called vertices V and a set of lines called edges E that connect some vertices together. The graph is defined as a set of vertices and edges that form a pair $G = (V, E)$.
- **Simple graph**: A graph in which each pair of vertices is linked by at the very most one edge and where no vertex has a loop.
- **Undirected Graph**: An undirected graph G is a pair (V, A) where V is a finite set of vertices and A is a set of unordered pairs of vertices. Also, loops are not allowed in undirected graphs.
- **Cycle**: A chain whose start and end nodes are the same and which does not use the same link more than once.
- **Hamiltonian Path**: A path that passes once and only once through each of the vertices of an undirected graph.
- **Hamiltonian Circuit**: simple cycle passing through all the vertices of a graph one and only once.
- **Adjacency Matrix**: Let G be an undirected graph with m vertices from 1 to m. We call the adjacency matrix of graph the matrix $A = (a_{jk})$ where a_{jk} is the total number of edges joining vertex j to vertex k:

$$\begin{cases} a_{jk} = w & \text{if and only if j and k are adjacent.} \\ a_{jk} = 0 & \text{if not.} \end{cases} \tag{1}$$

  with w is the weight of the edge (j, k).

- **Blum Blum Shub (BBS)**: is a pseudo-random number generator first proposed in [9]

$$x_{n+1} = x_n^2 \mod M \tag{2}$$

  With $M = pq$ the product of two large primes p and q.
  The complexity of the factorization of M is the main basis for the security of this generator, which means that the two primes must be carefully chosen to guarantee robustness.

III. Related Work

The application of graph theory in cryptography has become more emergent. However various encryption techniques have been proposed in this context.
A technique has been proposed by Amudha et al [10] that encodes clear messages through the Euler graph, the key used to protect the data in this approach is a kind of Hamilton circle. The authors in [11] sequentially construct three different graphs on the basis of an unconventional mapping, conjectured to be a one-way trapdoor function and designed specifically for graph structures. Some work focuses on the application of graph theory principles in computer networks and its potential to tackle the challenges of provisioning in secure cloud computing environments [12]. Two graph based public key cryptosystems have been suggested to secure sensitive Data in the work of Sensarma et al [13], where one is based purely on the properties of matrices, while the other is based on graph codes. In the work described in [14], the authors proposed a hybrid Cipher Block Chaining encryption system for e-mail protection. The suggestion was predicated on the integration of encryption technologies. Yousif et al [15], introduced a process to produce a new key on the basis of chaotic maps that are utilized to encode images. Within the work in [16], the emphasis is on the possibility of employing the Euler graph as a method object used in the remote method invocation (RMI) technique.

Among the most recent works, we mention the work presented in [17], where a block cipher system has been proposed using disjoint Hamiltonian circuits to present the data as a graph. Also in [18], a double vertex graph has been suggested to encrypt a word. At first, the given message was encrypted using the encryption table. The plaintext was then converted into a path graph. From the latter, a double vertex graph is constructed. We also mention the work [19], in which the original message is converted into several graphs. The ciphertext is obtained from the projection of the adjacency matrices representing the graphs into the secret key. A number of other proposals were suggested in the same thematic area [20], [21].

The originality of our work lies in the fact that our proposed system was able to blend both the concept of block ciphers, which is a major category of symmetric cryptography, and graph theory properties for representing plaintext, in particular Hamiltonian circuits, unlike the majority of works from the literature that rely only on graph theory properties to conceive their encryption systems.

IV. Proposed Approach

The primary aim that drives the system put forward in this paper, is to propose a robust variant of the encryption scheme proposed in [17] while maintaining the performance levels. The main concept on which our approach is based is inspired by the divide-and-conquer design method, which consists in dividing an initial problem into sub-problems and then addressing every component of the resulting subset independently. The final solution of the initial problem is then deduced from the solutions found to the sub-problems.

The system described in this work has been designed in such a way that it takes into account the complexity of

the processing that the plaintext messages are subjected to during their encryption process. Indeed, this is the objective of the contribution in this paper, which is to improve the processing of the plaintext by making it more difficult and more complex than [17], using mainly all the Hamiltonian circuits that represent the plaintext.

The scheme proposed in [17] used a block of 25-characters length, which can be represented by 2 disjoint Hamiltonian circuits in a graph of order 13, given that a graph of order 13 contains 6 disjoint Hamiltonian circuits (Theorem 1). In contrast to [17], which used only 2 of the 6 circuits, the concept put forward in this approach makes use of all the disjoint Hamiltonian circuits of the graph (6 circuits), which allows the representation of blocks with 78-characters length in a single graph.

Theorem 1: In a complete graph with n vertices there are $(n-1)/2$ edge-disjoint Hamiltonian circuits, if n is an odd number strictly greater than 3 [22].

Considering a message which consist of 78 characters, the formula for splitting into blocks in [17] would be as follows: $78 = 25 \times 3 + 3$. This means that four blocks will be transformed into four adjacency matrices. The formula used in the proposed algorithm is limited to a single block, which in turn will be partitioned into six sub-blocks to form a single graph with six disjoint Hamiltonian circuits, thus forming a single adjacency matrix (FIG. 1 illustrates the difference between both systems).

Generally, the pre-encryption process of the plaintext message involves several steps: First the plaintext is converted into ASCII values and then divides into several blocks of size 78 (referring by $Block_i$). This operation uses the following formula: $n = 78k+r$, where n is the size of the plaintext, r ($r \in [0,77)$] the remainder of the division of n over 78) represents the remainder of the plaintext after block partitioning, and k is the number of blocks (refers to the quotient).

$$\begin{cases} k' = k & \text{If the division is exact.} \\ k' = k+1 & \text{otherwise.} \end{cases} \quad (3)$$

Where k' represents the total number of blocks resulting from the division. Each $Block_i$ is partitioned into 6 sub-blocks of size 13(each sub-block is represented by $subBlock_{ij}$), which are then converted into Hamiltonian circuits where the weights of the edges of the graph G_i are represented by the ASCII values of the characters that compose them. Finally, the resulting graph is converted into an adjacency matrix M_i.

The main process involving in our proposed system are presented in the following:

A. Key Generation / Re-Generation Algorithms

The generation of the sub-key K_i occurs in four steps. The first involves the random selection of a character $Char$ from the $Block_i$. The second step consists in using the position corresponding to the ASCII value of $Char$ in two ways, to construct the vector of positions VP that is necessary for the decryption, as well as to recover the value N located in the same position in the master key KEK (of size 256), which will be used as the seed of the BBS generator. The third step

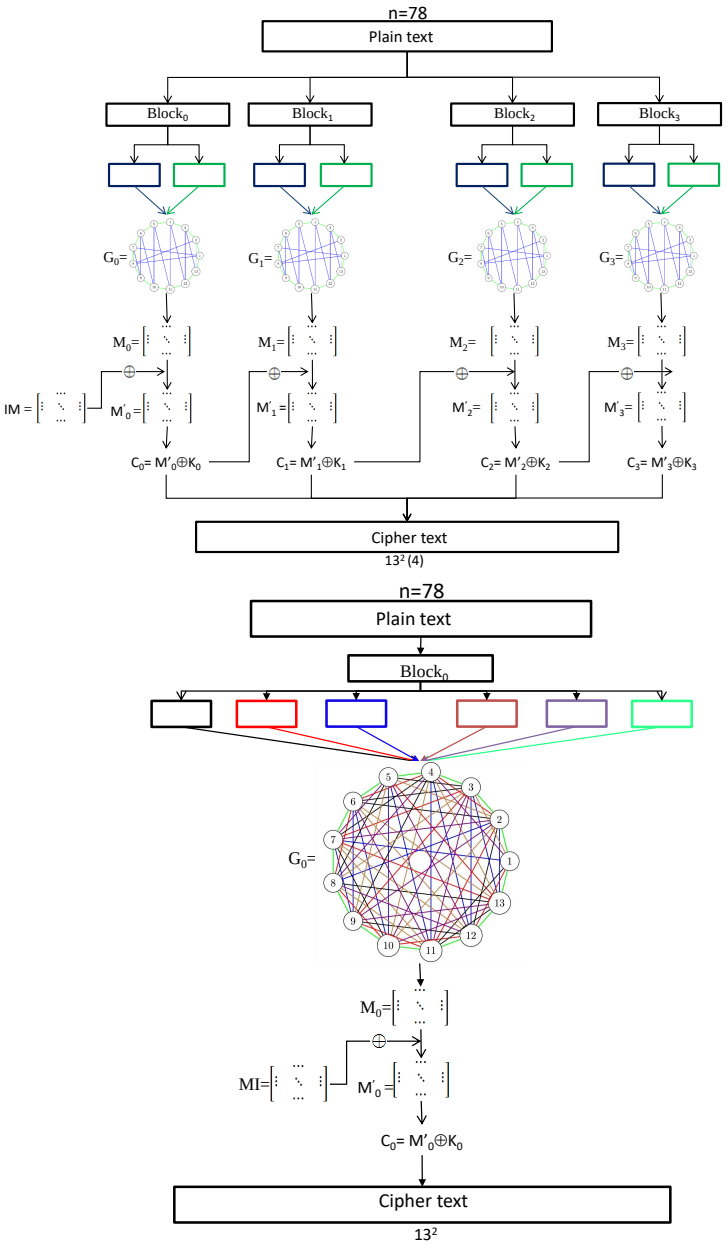

Fig. 1. Comparaison between the Encryption Process in [17] and the Proposed One.

in the process allows the generation of a vector S_i of size 13 from N using BBS generator. The fourth and final step uses the resulting S_i to generate the sub-keys K_i as a square matrix of order 13. The all sub-keys K_i(i = 0,. . ., k'-1) that are generated constitute the set $SK_{k'}$. This process is illustrated in FIG. 2.

The regeneration of K_i during the decryption process begins with the use of VP to recreate a key Key of size $13^2k'$ from KEK. Key is then divided into sub-vectors S_i of size 13 which are subsequently used to generate the sub-keys K_i as square matrices of order 13. This process is described in FIG. 3.

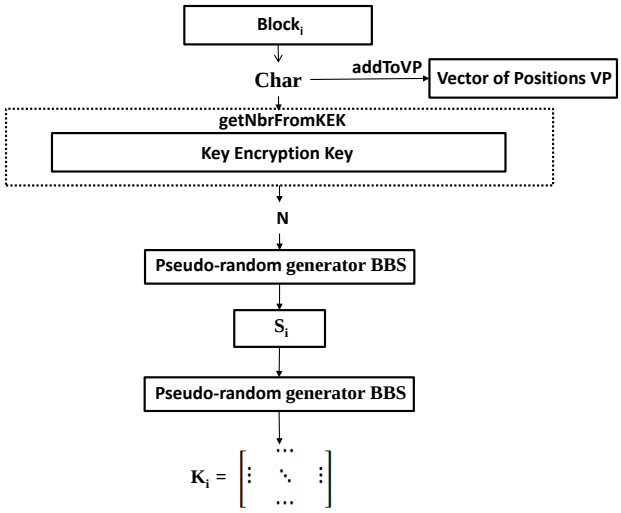

Fig. 2. Sub-keys Generator in Encryption Process.

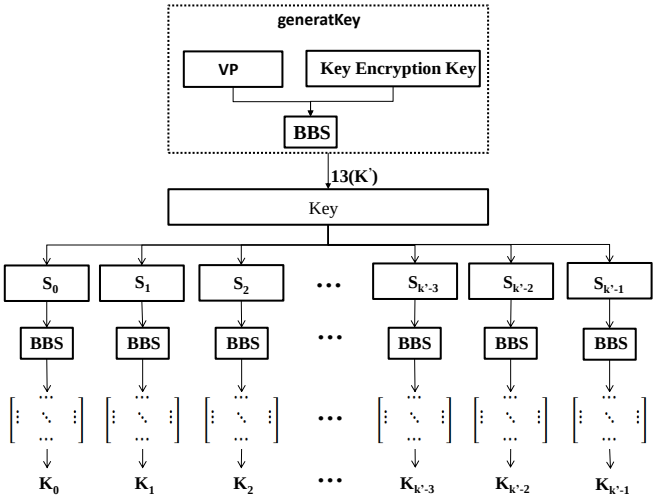

Fig. 3. Sub-keys Generator in Decryption Process.

Algorithm 1: Sub-keys generator in encryption process (GenerateSubKeys)

input : Clear message of n characters CMC_n , master key KEK, k' number of blocks
output: sub-keys $SK_{k'}$, the vector VP

begin
  /* Converts each character of the message into its ASCII value. */
  $CMA_n \leftarrow$ ***convertMessage***(CMC_n, n);
  /* Splits the message into k' $Block_i$ forming the set $BlockSet_{k'}$, where $BlockSet_{k'} = \{Block_0, \ldots, Block_{k'-1}\}$. */
  $BlockSet_{k'} \leftarrow$ ***parseMessage***(CMA_n, k');
  for *element* $Block_i$ *of the set* $BlockSet_{k'}$ **do**
    /* Randomly selects a character $Char$ from $Block_i$. */
    $Char \leftarrow$ **getCharFromBlock**$(Block_i)$;
    /* Feeds the vector VP with the ASCII value of the character $Char$. */
    $VP \leftarrow$ **addToVP**$(Char)$;
    /* Returns the content in the position p of the master key KEK, where p represents the ASCI code of the character concerned. */
    $N \leftarrow$ **getNbrFromKEK**$(KEK, Char)$;
    /* Generates from the seed N a vector S_i of size 13. */
    $S_i \leftarrow$ **generateSeed**(N, BBS);
    /* Takes as input the vector S_i and returns the sub-key K_i as a square matrix of order 13. */
    $K_i \leftarrow$ **generateSubKey**(S_i, BBS);
    /* Feeds the set $SK_{k'}$ with the sub-key K_i. */
    $SK_i \leftarrow$ **putSubKey**(K_i);
  end
end

B. Encryption Process

The encryption process begins with the pre-encryption step described above. Cipher block chaining (CBC) is used as mode of operation in our approach. The chaining uses a feedback method, in the sense that the result of the encryption of the previous block C_{i-1} is reused for the purpose of encrypting the current block M_i. More specifically, an exclusive or (XOR) operation is applied between the current block M_i and the previous block of ciphertext C_{i-1} as shown below:

$$M'_i = C_{i-1} \oplus M_i \tag{4}$$

A second XOR operation is then performed between the result of operation (3) and the sub-key K_i generated by the pseudorandom generator to compute the cipher C_i of the current block:

$$C_i = M'_i \oplus K_i \tag{5}$$

Since the first block does not have an antecedent. We generate an random matrix referring to IM(initialization matrix) which allows to perform the XOR operation with M_0. Each encrypted block consequently depends not only on the corresponding plaintext block, but also on all the encrypted blocks that precede it. The rows of the matrix C_i are concatenated to form a vector $eBlock_i$ of size 13^2, representing each encrypted block.

The resulting vectors $eBlock_i$ (i = 0,. . ., k'-1) generated from all blocks are then concatenated to form a single vector EM of size $13^2k'$. The encryption process, as shown in FIG. 4, ends with the transmission of the encrypted message EM in addition to the vector VP that is related to the decryption process.

Algorithm 2: Sub-keys generator in decryption process (GenerateSubKeys)

input : master key KEK, the vector of positions VP
output: sub-keys $SK_{k'}$

begin
  /* Generates a key Key of size 13k' from the vector of positions VP and the master key KEK. */
  $Key \leftarrow$ ***generateKey***(KEK, VP);
  /* Divides the key Key into k' vectors S_i (i = 0,..., k'-1). */
  $S_{k'} \leftarrow$ ***parseKey***(Key);
  for *element* S_i *of the set* $S_{k'}$ **do**
    /* Takes as input the vector S_i and returns the sub-key K_i as a square matrix of order 13. */
    $K_i \leftarrow$ **generateSubKey**(S_i);
    /* Feeds the set $SK_{k'}$ with the sub-key K_i. */
    $SK_i \leftarrow$ **putSubKey**(K_i);
  end
end

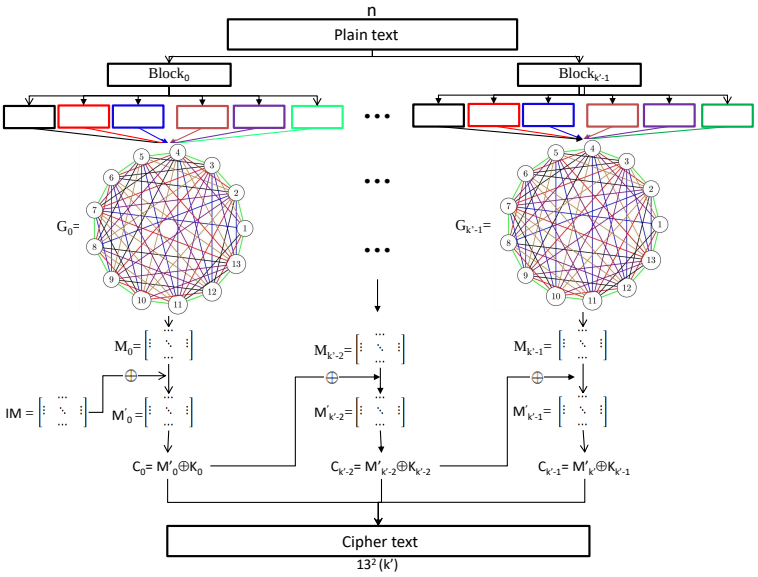

Fig. 4. Encryption Scheme.

Algorithm 3: Encryption Algorithm Using Disjoint Hamiltonian Circuits

input : Clear message of n characters CMC_n, master key KEK, Initialization Matrix IM of size 13
output: Encrypted message EM

begin
  $SK_{k'} \leftarrow$ ***GenerateSubKeys***(CMC_n, KEK, k');
  (IV-A).
  /* Converts each character of the message into its ASCII value. */
  $CMA_n \leftarrow$ ***convertMessage***(CMC_n);
  /* Splits the message into k' $Block_i$ forming the set $BlockSet_{k'}$, where $BlockSet_{k'} = \{Block_0, \ldots, Block_{k'-1}\}$. */
  $BlockSet_{k'} \leftarrow$ ***parseMessage***(CMC_n);
  for *element* $Block_i$ *of the set* $BlockSet_{k'}$ **do**
    /* Divides each block $Block_i$ into six sub-blocks $subBlock_{ij}$ (j = 0,...,5) of size 13, all forming the set $subBlockSet_i$, where $subBlockSet_i = \{subBlock_{i0}, \ldots, subBlock_{i5}\}$. */
    $subBlockSet_i \leftarrow$ **parseBlock**$(Block_i)$;
    /* Converts the sub-blocks into disjoint hamiltonian circuits in a graph G. */
    $G_i \leftarrow$ **blockToGraph**$(subBlockSet_i, 13)$;
    /* Transforms the graph G_i into an adjacency matrix M_i of order 13. */
    $M_i \leftarrow$ **graphToMatrix**(G_i);
    if *i=0* **then**
      $M'_0 \leftarrow IM \oplus M_0$;
    else
      $M'_i \leftarrow C_{i-1} \oplus M_i$;
    end
    $C_i \leftarrow M'_i \oplus SK_i$;
    /* Concatenates the rows of the matrix C_i to form the vector $eBlock_i$ of size 13^2. */
    $eBlock_i \leftarrow$ **transformMatixToVector**(C_i);
  end
  /* Forms a single vector EM of size $13^2 k'$ by concatenating the resulting vectors $eBlock_i$ (i = 0,..., k'-1). */
  $EM \leftarrow$ **concatenateEncryptionBlock**$(eBlock_{k'})$;
end

C. Decryption Process

In general, the process of decryption corresponds to the process of encryption performed in reverse order (ALGORITHM 4). In the decryption process described in this paper, the ciphertext EM refers to the input of the algorithm. EM is decomposed into k' vectors ($eBlock_i$) and then gathered to constitute the set $eBlockSet_k$. The $eBlock_i$(i = 0,. . ., k'-1) are subsequently transformed into a matrix C_i. The number of blocks k' is calculated as follows:

$$k' = m \div 13^2 \quad (6)$$

with m is the size of the ciphertext.

The sub-key generation algorithm presented in ALGORITHM 2 makes use of the provided vector VP to produce a key of size $13^2 k'$ from the master key KEK. Each block C_i(i = 0,. . ., k'-1) is decrypted using its own sub-key K_i using the following formula:

$$M_i = C_{i-1} \oplus M'_i \quad (7)$$

Where

$$M'_i = C_i \oplus K_i \tag{8}$$

and

$$M_0 = IM \oplus M'_0 \tag{9}$$

At this stage, the decrypted blocks M_i are transformed into a graph G_i and then into $Block_i$. Finally, the plaintext message is formed by concatenating the $Block_i$(i = 0,. . ., k'-1) as shown in FIG. 5.

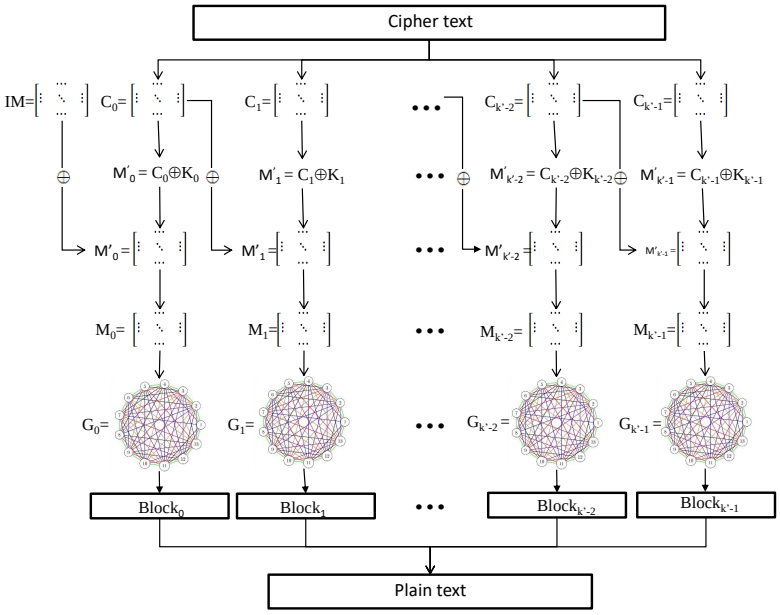

Fig. 5. Decryption Scheme.

Algorithm 4: Decryption Algorithm Using Disjoint Hamiltonian Circuits

input : Encrypted message EM, master key KEK, the vector of positions VP
output: Clear message of n characters CMC_n

begin
  $SK_{k'} \leftarrow$ ***GenerateSubKeys***(KEK, VP);
  (IV-A).
  /* Divides the encrypted message into k' $eBlock_i$ wich are then concatenate to form a set $eBlockSet_{k'}$. */
  $eBlockSet_{k'} \leftarrow$ ***parseEncryptedMessage***(EM);
  for *element* $eBlock_i$ *of the set* $eBlockSet_{k'}$ **do**
    /* Form the matrix C_i of order 13 from the vector $eBlock_i$. */
    $C_i \leftarrow$ **transformVectorToMatrix**$(eBlock_i)$
    $M'_i \leftarrow C_i \oplus K_i$;
    if $i = 0$ **then**
      $M_0 \leftarrow IM \oplus M'_0$;
    else
      $M_i \leftarrow C_{i-1} \oplus M'_i$;
    end
    /* Transforms the adjacency matrix M_i into a graph G_i. */
    $G_i \leftarrow$ **matrixToGraph**(M_i);
    /* Returns the $Block_i$ represented by the disjoint hamiltonian circuits inside the graph G_i. */
    $Block_i \leftarrow$ **graphToBlock**(G_i);
  end
  /* Forms a single block that forms the plaintext message by concatenating the resulting blocks $Block_i$ (i = 0,..., k'-1). */
  $CMC_n \leftarrow$ **concatenateBlock**$(Block_{k'})$;
end

V. Security Analysis and Experimental Results

The evaluation of the encryption system addressed the reliability of the suggested algorithm. For this purpose, we study the system reaction in terms of performance and security according to fundamental criteria. For which we also perform different statistical tests. This evaluation is described in the following sections.

A. Confusion and Diffusion Tests

Diffusion and confusion are very important as aspects of the functioning of a secure encryption which were first identified in 1949 by Claude Elwood Shannon [23]. In his original definitions:
Confusion means making the relationship between key and ciphertext as complicated and as involved as is feasible, whereas in this case refers to the property that redundancy in the plaintext's statistics is "dissipated" in the ciphertext's statistics.
Diffusion is related to the reliance of the output bits upon the input bits. In a cipher with proper diffusion, the changing of an input bit is expected to change every output bit with a probability of half (this is referred to as the strict avalanche criterion). Accordingly, the used equation (10) is:

$$bits_{diff} = (1 \div (13^2 \times 16) \times w(C \oplus C') \tag{10}$$

$$= (1 \div (2704) \times w(C \oplus C') \tag{11}$$

Where w is the hamming weight, C and C' are respectively the original and modified inputs, and the value 16 refers to the number of bits representing each element in the cipher.

B. Plaintext Sensitivity Test

The diffusion property is intended to produce an avalanche effect [24] between the plaintext and the encrypted messages. The sensitivity test of the bit change in the plaintext is used to verify the diffusion property of a particular algorithm.
Given pairs of plaintext and secret keys, we generate the ciphertext corresponding to each pair (plaintext, secret key) through our cryptosystem, changing one or more bits (Knowing that a change at character level implies a change of bit) in the randomly generated plaintext, and by retaining the key unchanged.
Subsequently, we calculate the average of the percentage of

bit difference by the equation (10) as illustrated in the FIG. 6. Over 50% of the bits in the cipher text are changed. We can clearly see that the average of the percentages of bit difference is between 48.16% and 51% for our encryption system and between 47.1% and 50.80% for AES-128. These percentages demonstrate that our encryption system offers a good diffusion compared to AES-128.

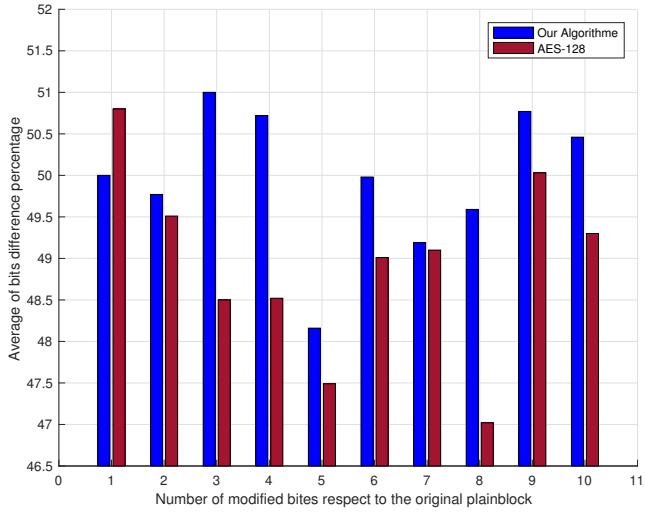

Fig. 6. Number of Modified Bits with respect to the Original Plaintext.

C. Key Sensitivity Test

The confusion property establishes a relation between the key and the ciphertext. The key sensitivity test ensures this property. Indeed, we consider a set of pairs of plaintext and secret keys. Each pair is encrypted by applying the proposed cryptosystem. Then, we modify one or more bits in the different randomly generated keys while the clear text still fixed. Afterward we calculate the average of the percentage of bit difference by applying the equation (10).

FIG. 7 represents the results obtained by using our encryption system and our generator to produce the encryption keys. We can notice that more than 50% of the bits are modified. Specifically, the average percentage of bit difference is between 49.64% and 50.79% for our encryption system and between 48.25% and 50.73% for AES-128. Thus, according to the experimental results, it can be said that the key generation via our algorithm is more robust than AES-128.

D. Statistical Tests

In order to study the quality of the random generation of the suggested encryption block cipher, we apply the well-known DIEHARD test [25]. The primary objective of this test is to demonstrate whether our cryptosystem is able to withstand statistical attacks. In other terms, the output of a secure block cipher must be indistinguishable statistically from a random output using the encryption function. To perform this test, a randomly generated cipher sequence is initially binarized to generate a bitstream of over 10 MB. Thereafter, the bitstream is analyzed statistically by putting it under the

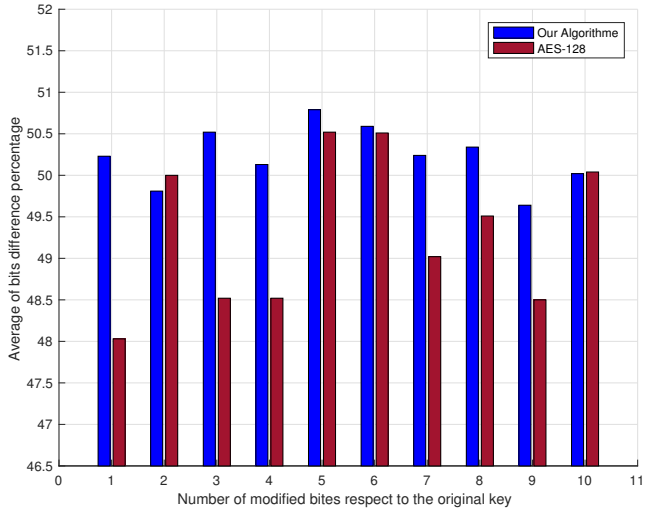

Fig. 7. Number of Modified Bits with respect to the Original Key.

DIEHARD tests. The DIEHARD tests check the p-value of the values generated randomly, with the p-value in the range [0.025, 0.975]. The average values found are summed up in TABLE I. The results indicate that the bitstream generated with our encryption system successfully passed all DIEHARD tests. Moreover, our encryption system shows a satisfactory and statistically indistinguishable random behavior.

E. Brute Force Attack

Brute force attacks are a mean to get all possible key arrangements with a fast prediction tool. Assuming that a high-performance machine that spends 10^{-10} seconds on testing the validity of every key is used, and that the numbers entailed in the master key range from 1 to 1000.
Given that algorithm has 1000^{256} potential keys. A brute force attack would require about $10^{-10} \times 1000^{256}$ seconds to find the appropriate key. Therefore, a brute force attack with an exhaustive search over the key possibility space is not feasible in a reasonable amount of time.

To find a 78-character message when a single block is used, it normally takes 1000 possibilities to find one of the master key numbers, which will represent the seed of the BBS generator involved in producing the S_0 vector. However, the prime numbers used as the input parameters for the generator are not easily determined (due to factorization problems). As a result, it is nearly impossible to figure out the sub-key if the pq product is large enough.

F. Time Analysis

Table II represents the performance test of our cryptosystem, compared to other known block ciphers such as triple DES [26] and AES [27] in terms of their CPU time consumption. The computations are run on a computer with on an Intel Core i7-6600U processor, 64-bit OS, 2.81 GHz with 20 GB of RAM. It can be seen from TABLE II that our algorithm can achieve good results in terms of run time over the other standard encryption systems.

TABLE I. DIEHARD TEST

Test Name	P-value	Interpretation
diehard bitstream	0.59537390	
diehard squeeze	0.97442749	
diehard sums	0.11133210	
diehard count 1s str	0.60934773	
diehard count 1s byt	0.78478421	
diehard parking lot	0.55915630	
diehard birthdays	0.03222200	
diehard operm5	0.75636037	
diehard oqso	0.33566335	
diehard dna	0.45051943	
diehard 2dsphere	0.53656799	
diehard 3dsphere	0.62980562	
diehard rank 32x32	0.40775458	
diehard rank 6x8	0.45554634	
diehard opso	0.44037399	
diehard runs	0.86351847	PASSED
diehard craps	0.15275419	
rgb bitdist	0.69014502	
rgb minimum distance	0.57113046	
rgb permutations	0.60422228	
rgb lagged sum	0.60927830	
rgb kstest test	0.26054914	
dab bytedistrib	0.68169231	
dab dct	0.25149694	
dab filltree	0.88848873	
dab filltree2	0.29185197	
dab monobit2	0.74899931	
sts monobit	0.68441660	
sts runs	0.37246909	
sts serial	0.50145101	
marsaglia tsang gcd	0.47467308	

TABLE II. ENCRYPTION TIME COMPARISON BETWEEN OUR BLOCK CIPHER AND OTHERS BLOCK CIPHERS USING DIFFERENT MESSAGE SIZE

Message Size (Kilo Byte)	AES (ms)	3DES (ms)	Our encryption algorithm
3	248.07	247.47	4.9
10	951.2	614.9	10.4
20	1972	1096	21.2

VI. CONCLUSION AND FUTURE WORK

The work presents a new cryptosystem that takes advantage of the principles of graph theory, which enable a high degree of security while maintaining the performance of data processing. Our proposed encryption block cipher using in particular the disjoint Hamiltonian circuits that have been adopted to represent the plaintext in a pre-encryption phase. the process makes use of a specific sub-key generator that has been set up to generate the encryption keys according to the requirements of the proposed system. We have performed different statistical tests, specifically the DIEHARD, confusion and diffusion tests to prove the security and performance of our cryptosystem. The experiments results proved the good behaviour of our proposed design in terms of robustness and CPU time compared to 3DES and AES. In a future work, we intend to use another pseudo-random generator, such as the one proposed in [28] known as PSOCA, which is mainly based on cellular automata, and we also investigate other properties of graph theory for a more discriminating and robust representation of the data.

REFERENCES

[1] A. J. Menezes, J. Katz, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[2] P. FIPS, "81, des modes of operation," *Issued December*, vol. 2, p. 63, 1980.

[3] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pp. 19–22, 2001.

[4] W. Meier, "On the security of the idea block cipher," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 371–385.

[5] N. P. Smart, "The "naive" rsa algorithm," in *Cryptography Made Simple*. Springer, 2016, pp. 295–311.

[6] ——, "Public key encryption and signature algorithms," in *Cryptography Made Simple*. Springer, 2016, pp. 313–347.

[7] A. J. Menezes, J. Katz, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[8] G. Alexanderson, "About the cover: Euler and königsberg's bridges: A historical view," *Bulletin of the american mathematical society*, vol. 43, no. 4, pp. 567–573, 2006.

[9] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on computing*, vol. 15, no. 2, pp. 364–383, 1986.

[10] P. Amudha, A. C. Sagayaraj, and A. S. Sheela, "An application of graph theory in cryptography," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 13, pp. 375–383, 2018.

[11] S. G. Akl, "The graph is the message: design and analysis of an unconventional cryptographic function," in *From Parallel to Emergent Computing*. CRC Press, 2019, pp. 425–442.

[12] K. D. Rangaswamy and M. Gurusamy, "Application of graph theory concepts in computer networks and its suitability for the resource provisioning issues in cloud computing-a review." *J. Comput. Sci.*, vol. 14, no. 2, pp. 163–172, 2018.

[13] D. Sensarma and S. S. Sarma, "Application of graphs in security," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 2273–2279, 2019.

[14] S. H. Hashem, "Proposal hybrid cbc encryption system to protect e-mail messages," *Iraqi Journal of Science*, vol. 60, no. 2, pp. 157–170, 2019.

[15] A. Yousif and A. H. Kashmar, "Key generator to encryption images based on chaotic maps," *Iraqi Journal of Science*, vol. 60, no. 2, pp. 362–370, 2019.

[16] T. A. Khaleel and A. A. Al-Shumam, "A study of graph theory applications in it security," *Iraqi Journal of Science*, vol. 61, no. 10, pp. 2705–2714, 2020.

[17] K. Bekkaoui, S. Ziti, and F. Omary, "A robust scheme to improving security of data using graph theory," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 5, 2020.

[18] C. Beaula and P. Venugopal, "Cryptosystem using double vertex graph," *Indian Journal of Science and Technology*, vol. 13, no. 44, pp. 4483–4489, 2020.

[19] P. Perera and G. Wijesiri, "Encryption and decryption algorithms in symmetric key cryptography using graph theory," *Psychology and Education Journal*, vol. 58, no. 1, pp. 3420–3427, 2021.

[20] S. G. Akl, "How to encrypt a graph," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 6, pp. 668–681, 2020.

[21] P. Venugopal, "Encryption using double vertex graph and matrices," *Solid State Technology*, vol. 64, no. 2, pp. 2486–2493, 2021.

[22] N. Deo, *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2017.

[23] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[24] J. C. H. Castro, J. M. Sierra, A. Seznec, A. Izquierdo, and A. Ribagorda, "The strict avalanche criterion randomness test," *Mathematics and Computers in Simulation*, vol. 68, no. 1, pp. 1–7, 2005.

[25] G. Marsaglia, "Diehard test suite," *Online: http://www. stat. fsu. edu/pub/diehard*, vol. 8, no. 01, p. 2014, 1998.

[26] D. Coppersmith, D. B. Johnson, and S. M. Matyas, "A proposed mode for triple-des encryption," *IBM Journal of Research and Development*, vol. 40, no. 2, pp. 253–262, 1996.

[27] N. P. Smart and N. P. Smart, *Cryptography made simple*. Springer, 2016.

[28] C. Hanin, F. Omary, B. Boulahiat, and S. Elbernoussi, "Design of new pseudo-random number generator based on non-uniform cellular automata," *International Journal of Security and Its Applications*, vol. 10, no. 11, pp. 109–118, 2016.