

A Novel Secure Transposition Cipher Technique using Arbitrary Zigzag Patterns

Basil Al-Kasasbeh

Faculty of Computer Studies, Arab Open University (AOU), Riyadh, Kingdom of Saudi Arabia

Abstract—Symmetric cipher cryptography is an efficient technique for encrypting bits and letters to maintain secure communication and data confidentiality. Compared to asymmetric cipher cryptography, symmetric cipher has the speed advantage required for various real-time applications. Yet, with the distribution of micro-devices and the wider utilization of the Internet of Things (IoT) and Wireless Sensor Network (WSN), lightweight algorithms are required to operate on such devices. This paper proposes a symmetric cipher based on a scheme consisting of multiple zigzag patterns, a secret key of variable length, and block data of variable size. The proposed system uses transposition principles to generate various encryption patterns with a particular initial point over a grid. The total number of cells in the grid and its dimension are variable. Various patterns can be created for the same grid, leading to different outcomes on different grids. For a grid of n cells, a total of $n! * (n-1)!$ total patterns can be generated. This information is encapsulated in the private key. Thus, the huge number of possible patterns and the variation of the grid size, which are kept hidden, maintain the security of the proposed technique. Moreover, variable padding can be used; two paddings with different lengths lead to a completely different output even with the same pattern and the same inputs, which improves the security of the proposed system.

Keywords—Cryptography; symmetric cipher; block cipher; transposition algorithms

I. INTRODUCTION

Data confidentiality can be ensured using both asymmetric cipher and symmetric cipher. Symmetric cipher has the advantage of low computational requirements compared to asymmetric cipher cryptography, which is commonly used for key exchange and authentication purposes [1]. Asymmetric cipher is commonly referred to as public-key encryption [2]. On the other hand, the symmetric cipher is commonly referred to as private-key encryption and has two main types: the block cipher and the stream cipher. The stream cipher encrypts each bit of the message with each key's bit using operations, such as the exclusive-or (XOR) [3].

On the other hand, block cipher uses substitutions and transposition operations on an n -bits block, as illustrated in Fig. 1. Block cipher encrypts a data block of predetermined length using private key principles. Both stream and block ciphers have their applications, advantages, and disadvantages. The stream cipher is fast and requires low computational power compared to the block cipher. Block cipher is widespread and is used in various other applications, such as stream cipher, hash function, pseudorandom number generator, and message authentication [4]. Besides, the block cipher is more secure than the stream cipher, subject to the strength of the utilized

private key [5]. The security level of the block cipher algorithm is evaluated based on the complexity, the performance, and its strength against possible cryptanalyses, such as linear and differential analysis and the homegrown crypto that is created afterthought [4, 6, 7].

Block cipher algorithms use predefined block sizes. A multiple of 8-bits block sizes is commonly utilized as compatible with most processors. Besides, the large size is avoided as it leads to padding, increasing computational requirements. Similarly, small size blocks give more chances for dictionary attacks on the ciphertext blocks to succeed [8]. For a variable-length message to be encrypted, the message is first divided into blocks of the predetermined size, padded if necessary to meet the required size. Then each block is encrypted using the cipher algorithm with a private key. Advanced algorithms for block cipher have been developed, such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES), which are the most utilized encryption algorithms worldwide [4, 9]. These algorithms depend on mathematical operation and substituting the plaintext bits and characters by other bits and characters; these are called substitutional algorithms. Besides the substitution cipher, another block cipher type is called transposition cipher, such as rail fence and columnar. Compared to the substitution cipher, the transposition cipher is faster, as it depends on transposition shuffle rather than mathematical operations. Yet, because the shuffle is limited and known in advance, and the algorithms are operated on very small keys, the security of these algorithms is weak. Surprisingly, vast number of the existing substitutional algorithms has been solved by the cryptanalysis [4, 5, 10, 11].

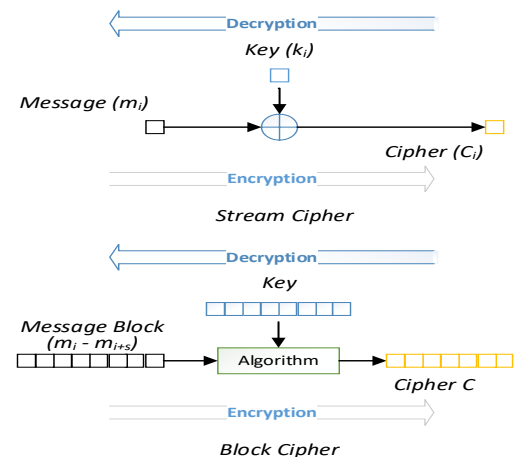


Fig. 1. Stream Cipher vs. Block Cipher.

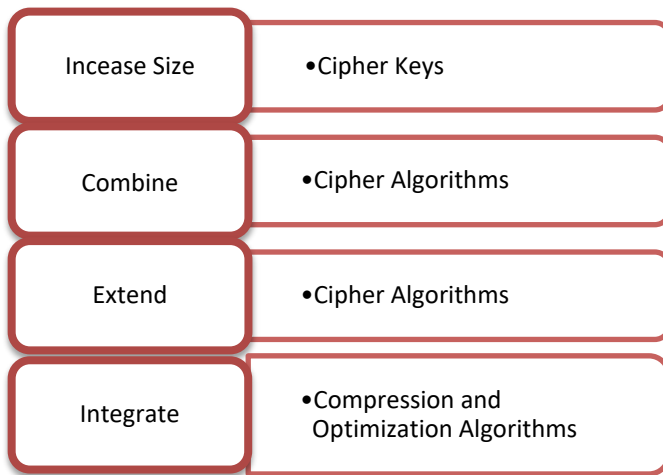


Fig. 2. Modern Cryptograph Directions.

Accordingly, various modern cipher techniques have been proposed to improve the security of the block cipher. Various ways have been followed to extend the existing algorithms. As illustrated in Fig. 2, these are 1) Proposing various encryption modes, in which the block ciphers are related to each other or related to randomly created vectors to increase the complexity of the generated cipher. 2) Combining various cipher algorithms to create a more secure system. 3) Combining cipher algorithms with optimization and compression algorithms. 4) Extending the cipher algorithms by increasing the size of the key or modifying their structure. Although these directions improve the security of the cipher algorithms, most of them have increased the computational requirements of the encryption and decryption process. The complexity of the modern techniques for data encryption leads to the inapplicability of these techniques to be utilized in most of the modern applications, such as real-time applications, sensor networks [12], and Internet-of-Things (IoT) [13], which demands secure yet highly performance encryption techniques [14].

Accordingly, this paper proposes a new technique for block cipher encryption. The proposed system follows different directions compared to these mentioned previously in improving the security of the symmetric cipher cryptography [15]. The goal of the proposed technique and the creative direction is to improve security while maintaining low encryption and decryption processes requirements. The proposed technique improves the security using different transposition zigzag patterns while solving one of the major transposition cipher problems: the limited possible shuffle operation of the cipher algorithm. Moreover, the proposed technique used variable grid size maintained secret in the private key, similar to the columnar algorithm. Thus, before the encryption is made available for the sender, various patterns are created and saved with their identification number in the system. These patterns are shared publically between the sender and the receiver. However, trying all these patterns is impossible for cryptanalysis because of the vast possible patterns. Then, the proposed system allows the sender to freely choose among different patterns and select the grid size on

which the patterns are drawn. The size and the pattern form the private key of this cipher system. A variable-length message can be divided into blocks of various lengths, padded if necessary to meet the required length. Each block is encrypted using different patterns and keys.

II. LITERATURE REVIEW

Transposition ciphers are implemented based on designed patterns to scramble the plaintext. Examples of some of the transposition ciphers are presented in Fig. 3. The rail fence used the number of rows as the key. Then, a grid is created with the specified number of rows and number of columns equal to the length of the message to be encrypted. The plain text is placed diagonally downwards on successive rails on an imaginary fence, followed by upwards moves as the movement reaches the grid's last row. The letters are read row by row to create the ciphertext [16]. In the columnar cipher, the key determines the number of columns and the columns' order. Then, a grid is created with a specific number of columns and rows to accommodate all the letters in the message to be encrypted. The plain text is placed on the grid row by row and in order. Then, the columns are permuted based on the key. The letters are read column by column to create the ciphertext [16-19].

The route cipher is another grid-based transposition cipher with longer keys than the rail fence and columnar. The key represents the number of columns or rows and the patterns to read the letters in the grid. The plaintext is written in the grid row by row and in order. Then, the ciphertext is produced by reading the grid upwards or downwards clockwise or zigzag patterns up and down. Double or multiple columnar ciphers are used to improve the security of the single columnar cipher. The same key can be applied, or other keys are required in multiple rounds of the columnar cipher process [18]. Myszkowski is a variation of the columnar cipher with a key of repeated letters that are given the same order. Accordingly, columns with unique numbers are read downward, and columns with recurring numbers are transcribed left to right. Disrupted transposition is another grid-based cipher with irregular spaces added between the plaintext letters. Overall, various transposition ciphers have been proposed. These transposition ciphers depend on a grid to scramble the plaintext. Yet, the problem of these ciphers is the limited patterns with a small key space that can be discovered in the cryptanalysis processes that are searching the keyspace [20].

Various modern techniques were proposed by modifying the transposition cipher algorithm. Sokouti, Sokouti [21] added 8-bits padding to each 8-bits in the message, structured these bits in a binary tree, and traversed this tree using an in-order traversal algorithm. However, such complex processes increase the complexity of the transposition cipher. Twum, Hayfron-Acquah [22] extended columnar cipher with a modified Rubik's cube puzzle with a higher level of security as the cube represents a 3-dimensional instead of the 2-dimensional space utilized in the original columnar cipher. The number of such modified algorithms is enormous. Block cipher surveys were presented by Surya and Diviya [23], Albermany and Radhihamade [10], and Mandal [24].

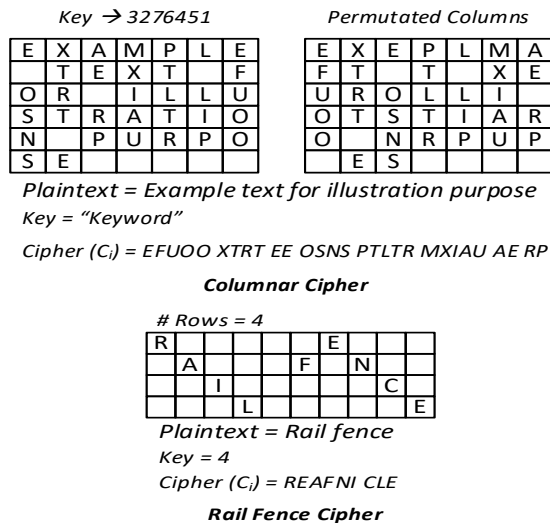


Fig. 3. Example Transposition Ciphers.

Lasry, Kopal [18] extended columnar cipher with a two-phases hill-climbing algorithm in combining cipher algorithms with optimization. The main goal of the developed algorithm is to increase the size of the key to improve the security of the cipher. Hill climbing is an optimization algorithm that starts with a random solution and iteratively finds optimal. With columnar cipher, hill climbing is used to swap columns and increase the adjacency score of the resulting cipher. Various other schemes combined cipher algorithms with optimization algorithms, such as genetic algorithm (GA) [25, 26]. Using the Fibonacci code algorithm, Siregar, Fadlina [19] integrated columnar cipher and data compression. The columnar output is input to the Fibonacci algorithm, which produces the final ciphertext.

Encryption modes describe how the blocks of a single message are related to each other to improve the security of the message. The electronic codebook (ECB) encryption mode for block cipher encrypts each block individually. The advantages of this mode are the ability of out-of-order decrypting, non-propagation of errors between blocks, speed, and parallelism. Yet, as the same key is utilized, the same input plaintext produces the same cipher, which is deterministic and can be attacked through traffic analysis [4]. To overcome this limitation, several encryption modes have been developed; these are cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and counter (CTR) mode [11]. These modes operated based on the concept of using extra data besides plaintext and the key for creating probabilistic encryption. CBC encrypts the first block together with a randomly initialized vector. The vector is combined with plain text before the encryption process. The ciphertext is then used with the second block in place of the random vector, and the output is used with the third block and so on. In CFB, the random vector is encrypted and combined with plaintext, which emulates the self-synchronizing stream cipher. OFB is similar to CFB except that the encrypted vector is sent to the next block instead of the ciphertext, similar to the synchronous stream cipher concept. CTR mode initializes a unique vector for each block, encrypts the vector then combines the output

with the plaintext to produce the ciphertext [27]. These modes aim to scramble the output, avoid repeated patterns, and provide semantic security, to prevent inference of any information from the ciphertext under an unknown key [4, 11]. Yet, these modes create various disadvantages, mainly related to the computational power and the required processing time. Generally, these modes have been developed with substitution encryption in mind. Although these modes applied for transposition, less attention has been given to this encryption scheme, mainly because of the limitation related to its weak security, resulting from the weakness of their keys, and the limited variation of the transposition shuffle [28-30].

In the integration direction, Srikantaswamy and Phaneendra [31] integrated columnar transposition cipher with Caesar substitution cipher. The secret key is generated randomly based on a selected seed value. Then, Caesar cipher is implemented with alphabets, symbols, and numbers as an extension to the original version that uses alphabets only. Finally, columnar is implemented using a randomly selected number of columns. Kester [17] integrated columnar transposition cipher with Vigenere substitution cipher. Columnar is implemented using a randomly selected number of columns. The generated ciphertext is then used as the key for encrypting the plaintext using the Vigenere cipher. Dar [16] integrated columnar cipher, Caesar cipher, and rail fence cipher to improve the security of the ciphertext. The aforementioned cipher algorithms are implemented in order; as such, the output of the columnar is used as input to Caesar, and the output of Caesar is used as input to the rail fence, which produces the final ciphertext.

As noted, these extended techniques rely on more processing rounds and complex calculation and transposition to secure the data. Yet, those techniques cannot be applied for micro-devices with limited resources or real-time applications requiring rapid encryption-decryption processes [32]. Accordingly, this research proposed a strong security technique resulting from the possible variation with low processing requirements.

III. THE PROPOSED SYSTEM

The proposed technique relies on predefined zigzag patterns drawn on a two-dimensional grid and can be extended to three-dimensional as required. The patterns can be created manually, as illustrated in Fig. 4, or automatically using graph-based search, or randomly depending on the application and the device. In a typical implementation of the proposed techniques, these patterns are made publically available and exchanged between the sender and the receiver. Yet, to make the system secure, the number of these patterns should be large enough to avoid brute-force solving of the ciphertext. Moreover, some patterns can be made secret and shared only between the sender and the receiver.

Besides, these patterns are saved in the sender and receiver devices in two arrays, one for the x-axis and the other for the y-axis, as illustrated in Fig. 5. If a three-dimensional cube is utilized, another array is used for the z-axis. Such representation requires low memory and can be traversed with O(n) complexity, where n is the size of the grid. The vast number of possible patterns maintains security. For a grid of n

cells, a total of $(n!)$ can be generated. Moreover, given that n is variable, $(n! * (n-1!))$ total patterns can be generated if n is hidden. The value of n can be made invisible as the proposed technique allows for arbitrary padding at the end of the message before and after encryption. For example, for a message of length 6, 720 $(6!)$ patterns are drawn on a grid of size 6 $(2*3)$. Yet, after encryption padding, the message can appear of size 9, for example. Thus, for cryptanalysis to try all possible patterns, all patterns of the grid of sizes $\{9, 8, \dots, 1\}$ should be examined.

The line-crossing patterns can be created manually but preferably using random number generation. The grid size and the initial cell s are determined to generate a manual pattern. Then, the pattern is created following zigzag directions visiting all the cells once and terminating the traversing at the arbitrary final cell, e . The coordinates of the cells are preserved in the arrays based on the traversing order. Each pattern is saved with a unique identifier to reference between the sender and the receiver. Besides, a generated pattern can be kept secret and exchanged with the private key because the pattern can be represented as two arrays in a two-dimensional grid. Although the zigzag patterns required manual attention to avoid line-crossing, these zigzag patterns can be generated using a graph representation of the grid and graph traversals mechanisms, such as in-order, pre-order, and post-order, as illustrated in Fig. 6. Nevertheless, as illustrated in Fig. 7, line-crossing patterns do not affect the security of the proposed technique. Instead, such line-crossing patterns allow for random patterns without the need for human attention or graph representation and traversals.

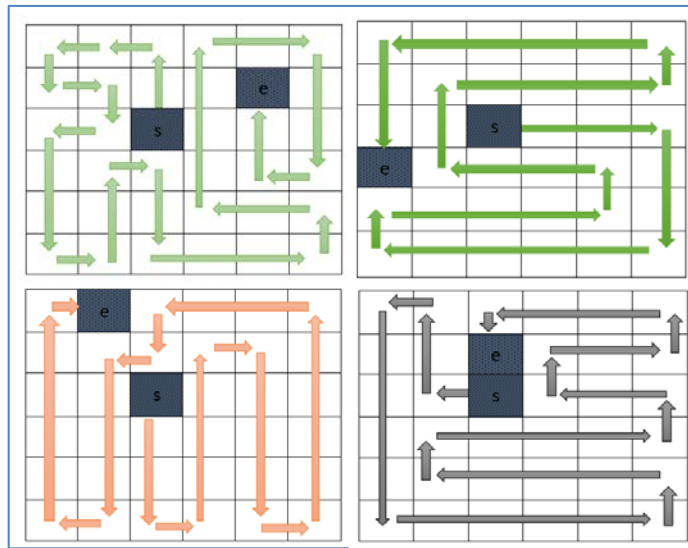


Fig. 4. Zigzag Patterns Examples.

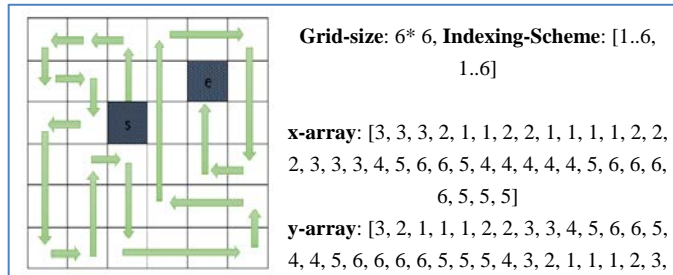


Fig. 5. Example of Array Representation of a Zigzag Pattern.

A. Pattern Generation

The pattern by which the transposition processes are implemented can be created in two forms and three ways. The zigzag patterns, which do not include any line-crossing, can be generated either manually or using graph-based techniques, as listed in Table I.

TABLE I. PATTERNS LIST OF THE PROPOSED TECHNIQUE

Pattern Type	Generated Mechanism	Advantages
Non-crossing Zigzag	Manual	Visually interpreted and verifiable.
Returnable Pattern	Graph Traversal (In-order, pre-order, and post-order)	Visually interpreted and verifiable and created automatically
Crossing-Line Pattern	Manual or Randomly	Easily created with wide varieties

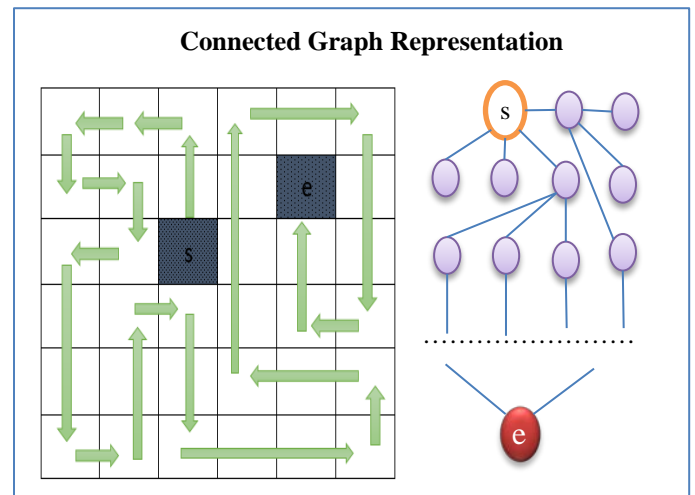


Fig. 6. The Graph Representation for a Zigzag Pattern Generation.

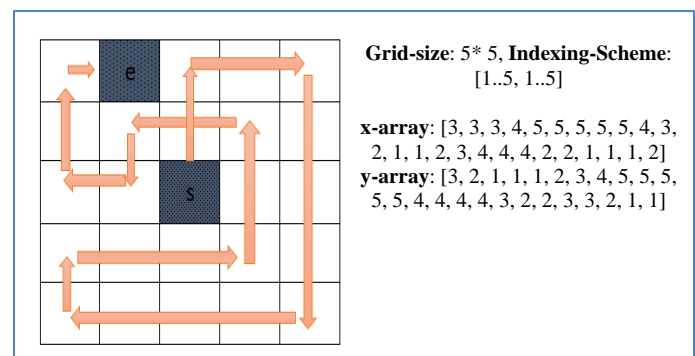


Fig. 7. Crossing-Line Pattern.

B. Secret Key Value

The secret or the private key of the proposed technique is represented with an integer value, which can be converted into binary as required. The private key of the proposed cipher technique referred to the grid size, the pattern to be utilized, and the padding options. There are two options for determining

the pattern, either a secret pattern embedded into the private key or the unique identifier of a public pattern. A secret pattern can be identified as starting by -1 value. The public pattern is identified with a unique positive identifier. The identifier can be of any length to continuously add patterns without limits. There are also two padding options to improve the security of the proposed technique, as listed in Table II. Table II lists the private key components in the proposed technique with examples.

TABLE II. PRIVATE KEY COMPONENTS OF THE PROPOSED TECHNIQUE

Components		Description	Example
Length		Four digits: The first digit indicates using public (0) or secret pattern (1). The next two digits indicated the length of the pattern (either length of the identifier or length of the included pattern). The next digit indicates whether padding is done before (0) or after encoding (1). If no padding occurs, then this digit can take any value.	0160 → using public pattern identified by 16 digits with padding implemented before encrypting
Grid size		4-digits for the two-dimensional	0610 → six rows and ten columns
Pattern	Public Identifier	Variable number of digits for the unique identifier	Can be any number
	Secret Pattern	Two concatenated arrays of size equal to the grid size (identified by the size field in the pattern)	21211212 → two concatenated arrays [2121] and [1212] that describe a pattern over 2*2 grid
Padding	Before encoding	Two digits to indicate the length of the padding at the end of the grid	05 → padding of length 5
	After encoding	Two digits to indicate the length of the padding at the end of the ciphertext	04 → padding of length 4
Example Keys			
1501 0505 3334555554321123444221112321112345555444432233211 06			
A secret pattern consists of 50 digits of the secret pattern or a 5x5 grid with six digits padding before encrypting			
0101 0505 0125214578 06			
A public pattern has an identity consisting of 10 digits with the same specification as the previous one			

C. Encryption

The encryption process is implemented based on the specifications stated in the secret key. First, the grid is created based on the predetermined dimensions, utilizing the secret key components. The plain text components are placed inside the grid based on the pattern represented by the x-array and the y-array. The ciphertext is produced by reading the grid row by row as the grid is filled. If padding is implemented before encoding, then the padded code (either 0's in case of encrypting binaries or X's in the case of encrypting letters) is considered in the last row(s) of the created grid. For example,

if a 3x3 grid is utilized with two padding bits before encoding, then the value of the cells $\{(2,3), (3,3)\}$, will be zeros, and these cells will not be filled with the components of the plaintext as will be explained. If padding is implemented after the encoding, the padded code is attached after the ciphertext is produced. Algorithm 1 presents the encryption process. The significance of the padding is that, even with the same pattern and the same inputs, different pad lengths for the before encoding padding leads to a completely different output.

Algorithm 1: Encryption

```

1 Input: mb, K (cols, rows, xs [], ys [], beforePadding, afterPadding)
2 Output: cb
3 Grid [[]]:= Create-Grid (cols, rows)
4 xPadding [],yPadding []:= IdentifyPaddingCells(beforePadding)
5 For-Each x,y in xs, ys
6     IF (x,y ∈ xPadding , yPadding)
7         Grid [x,y] := 0
8     ELSE
9         Grid [x,y] := mbi, i++
10 For-Each c in Grid
11     cbi := c
12 IF (afterPadding > 0)
13     Concatenate (cb, pad)
    
```

Notations:

mb, cb: message block and the cipher block, respectively
k: private key
cols, rows: number of columns, and rows as determined by the private key, respectively
xs, ys: the x-array and y-array of the pattern, respectively.
beforePadding, afterPadding: length of the padding before and after encryption, respectively.

As given in Algorithm 1, line 3 created the grid used by the encryption. The padded cells used before encryption are identified based on their numbers, as the last cells in the grid, as given in line 4. Lines 5-9 fill the grid following the pattern with the message (lines 8-9) or with padding (lines 6-7) if the pattern comes across the padding cell. Lines 10-11 create the ciphertext by reading the grid row by row. Finally, lines 12-13 concatenate after encryption pads if it exists as determined in the key.

D. Decryption

The grid is created based on the predetermined dimensions after receiving the encrypted message (bit sequence or letter sequence) at the receiver side. The grid is filled in a row by row manner, similar to how the sender created the ciphertext in the last stage. Then, to retrieve the plaintext, the receiver will implement the same pattern as utilized in the receiver side, but this time to read the cells and reproduce the message. The padded code cells are skipped while reading the grid components if padding is implemented before encoding. At the same time, if padding is implemented after encoding, the last components of the received ciphertext will be removed. Algorithm 2 presents the decryption process.

Algorithm 2 represents opposite operations. Line 3 created the grid. After encryption pads are removed, then in lines 5-6. Then, the grid is filled in lines 7-8. Lines 5-9 read the grid following the pattern and filling the message (lines 12-13) or skip padding (lines 10-11) if the pattern crosses the padding cell.

Algorithm 2: Decryption

```

1  Input: cb, K (cols, rows, xs [], ys [], beforePadding, afterPadding)
2  Output: mb
3  Grid [[]]:= Create-Grid (cols, rows)
4  xPadding [],yPadding []:= IdentifyPaddingCells(beforePadding)
5  IF (afterPadding > 0)
6    Eliminate (cb, pad)
7  For-Each c in Grid
8    cbi := c
9  For-Each x,y in xs, ys
10   IF (x,y ∈ xPadding , yPadding)
11     continue
12   ELSE
13     mbi := Grid [x,y], i++
    
```

Notations:

mb, *cb*: message block and the cipher block, respectively
k: private key
cols, *rows*: number of columns, and rows as determined by the private key, respectively
xs, *ys*: the x-array and y-array of the pattern, respectively.
beforePadding, *afterPadding*: length of the padding before and after encryption, respectively.

E. Multiple Encryption-Decryption

Multiple patterns can be used with the same block with multiple rounds while considering the computational requirements at the sender and the receiver, as given in Fig. 8. Accordingly, multiple private keys are required for such a purpose. Nevertheless, multiple rounds can be done with the same key, as each round will contribute to the scrambling of the input.

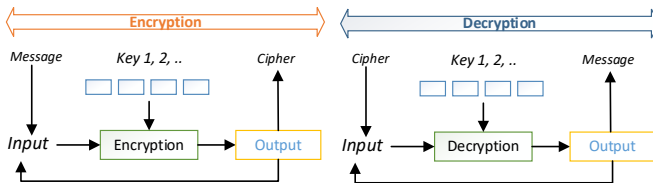
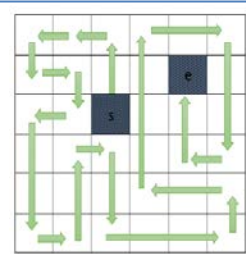


Fig. 8. Multiple Encryption-decryption.

IV. EXAMPLES

An example of encrypting and decrypting a plaintext represented using a binary code and a private key, which specify the size of the grid and the pattern, is given in Fig. 9. A single block is encrypted and decrypted using the given key in the example. The rest of the blocks can be encrypted and decrypted in the same way if the same key is utilized. Otherwise, different grid-size and different patterns are utilized.

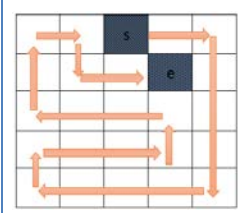


Key: 0101|0606|2025214515|00

Message:
100111001011010001010001011110100111

Cipher:
110110100111101110000010110100101010

Fig. 9. Example of Binary Encrypting and Decrypting.



Key: 0091|0505|145786952|00

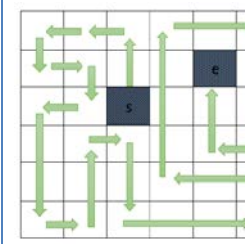
Message:
ABCDEFGHIJKLMOPQRSTUVWXYZ

Cipher:
VWABCUXYZDTSRQELMNPEKJIHG

Fig. 10. Example of Letters Encrypting and Decrypting .

As shown in Fig. 10, the second example represents the encrypting and decrypting of letters and numbers. The process is identical to the previous examples. Using different keys leads to using different grid sizes, different patterns, and different encoding padding. The letters in the example can be encrypted directly as given in the example or converted into binary code and encrypted bit-wise.

As given in Fig. 11, the third example represents the encrypting and decrypting of binary code before encoding padding. As noted, the results were contributing to more scrambling of the message.



Key: 0100|0606|2025214515|11

Message:
ABCDEFGHIJKLMOPQRSTUVWXYZ

Cipher:
EDCRSTFGBQZUIHAPYVKLMOXWK

Fig. 11. Example of Binary Encrypting and Decrypting with Padding.

V. SECURITY CONCERNS

Unlike the substitution cipher, attacks on transposition cipher do not depend on linear and differential cryptanalysis [33, 34]. Instead, the attacks on transposition cipher depend on guessing the key with statistical analysis of the n-gram of the language [35]. In such a process, the optimization algorithms reduce the time required by the brute-force approach. Accordingly, there are two strength issues to defeat such cryptanalysis: increasing the keyspace and increasing the possibilities of the transposition and scrambling processes. In the proposed technique, the key size has been increased to variable size, given that the size can be encapsulated into the private key itself. Although the size of the block can reveal much about the grid possibilities, using padding in two ways increases the block's size and leads to variable block size. Accordingly, padding the text with long pads are preferable to increase the security of the proposed technique. Moreover, as the encryption patterns are hidden in the private key, it is hard to implement all possible patterns to discover the correct one. Semantic security is granted in the proposed technique as it cannot infer any information from the ciphertext under an unknown key. To summarize, the security of the proposed technique can be listed as given in Table III.

The proposed system's validation is implemented based on various texts encrypted with random keys and with reference to the frequency distribution graph of the standard English letter, as illustrated in Fig. 12. This validation assumes that the

attacker captures the ciphertext without knowing the key. Accordingly, the attacker implements frequency distribution to analyze the captured ciphertext. An example of this graph is given in Fig. 13(a). The attacker then compares the frequency distribution graph with the Standard English Letter. Matching is then implemented by shifting the distribution graph to match the distribution graph of the English letters. As given in Fig. 13(b), the matching can be found by shifting the ciphertext graph by 3 letters. Shifting has been determined as the letter "E" is the most frequent in English, while letter H is the most frequent letter of the ciphertext graph, as it has been used in padding. Yet, the plaintext was not obtained as the reverse shifting is implemented. Accordingly, frequency analysis of 1000 different samples and shifting is determined by matching the frequency graph of each sample with the letter "E". The obtained results after shifting did not match any input text of these samples.

TABLE III. STRENGTH ASPECTS OF THE PROPOSED TECHNIQUE

Strength Aspect	Description
Vast patterns	The number of patterns that can be created is large
Variable grid size	Using padding, the actual size of the grid is hidden
Long key	The key length can be large with patterns included or long pattern identity
Hidden pattern	Secret patterns make the process of breaking the cipher harder

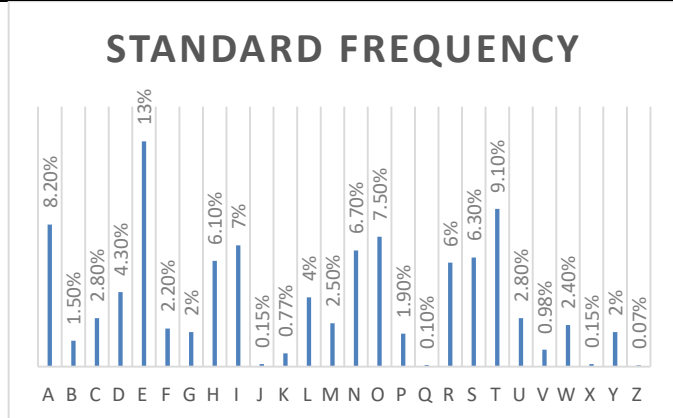
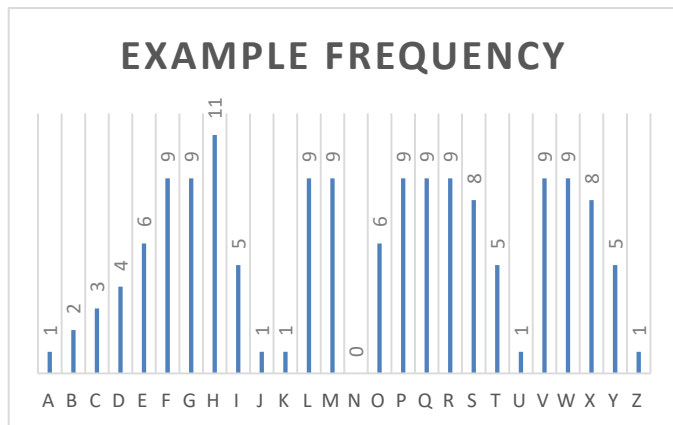
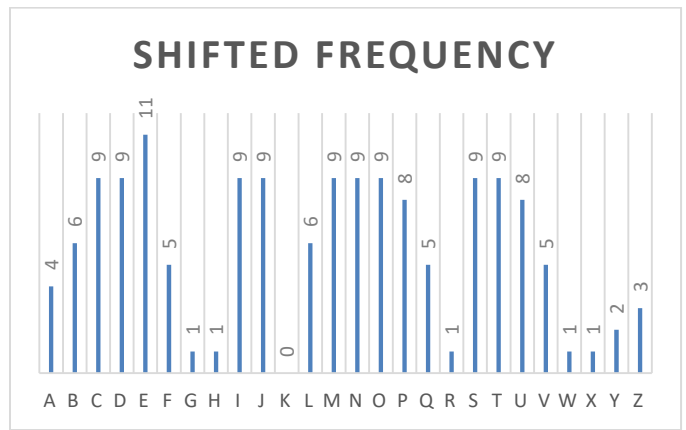


Fig. 12. Standard English Text Frequency.



(a) Example Frequency Graph.



(b) Example Frequency Graph after Shifting.

Fig. 13. Frequency Analysis of Ciphertext.

VI. CONCLUSION

In this paper, a technique for a transposition block cipher is proposed based on arbitrary grid size, initial point, and arbitrary zigzag patterns. The proposed technique improves the security of the ciphertext by scrambling the letters or the bits of the plaintext in an unpredicted manner. The time and memory requirements of the proposed technique are maintained as low as possible to cope with the requirements of currently utilized micro-devices and real-time applications. Accordingly, the proposed technique is of $O(n)$ complexity, where n is the grid's size. High security is maintained using a large key, data block of variable size with vast possible patterns. To defeat the cryptanalysis, the block size is increased using padding in two ways. Two similar paddings, yet with different lengths, lead to a completely different output even with the same pattern and the same inputs, which improves the security of the proposed system. Accordingly, hiding the patterns, the block size, and the padding in the private key makes it hard to implement all possible patterns to discover the correct one. The frequency analysis implemented proves the security of the proposed technique.

ACKNOWLEDGMENT

The author would like to thank Arab Open University-KSA and Oracle-KSA for supporting this study.

REFERENCES

- [1] N. Li, "Research on Diffie-Hellman key exchange protocol," 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 2010, pp. 634-637.
- [2] M.R. Joshi and R.A. Karkade, "Network security with cryptography," International Journal of Computer Science and Mobile Computing, vol. 4(1), 2015, pp. 201-204.
- [3] M. U. Bokhari, S. Alam, and F.S. Masoodi, "Cryptanalysis techniques for stream cipher: a survey," International Journal of Computer Applications, vol. 60(9), 2012.
- [4] D. Bujari, and E. Aribas, "Comparative analysis of block cipher modes of operation," International Advanced Researches & Engineering Congress, Osmaniye, Turkey, 2017, pp. 1-4.
- [5] Valea, E., et al., Stream vs block ciphers for scan encryption. Microelectronics Journal, 2019. 86: p. 65-76.
- [6] S. Rani and H. Kaur, "Technical review on symmetric and asymmetric cryptography algorithms," International Journal of Advanced Research in Computer Science, vol. 8(4), 2017.

- [7] A.G. Khan, S. Basharat, and M.U. Riaz, "Analysis of asymmetric cryptography in information security based on computational study to ensure confidentiality during information exchange," *International Journal of Scientific & Engineering Research*, vol. 9(10), 2018, pp. 992-999.
- [8] G. Singh, A. K. Singla, and K. S. Sandha., "Superiority of blowfish algorithm in wireless networks," *International Journal of Computer Applications*, vol. 44(11), 2012, pp. 23-26.
- [9] S. Singh, S.K. Maakar, and S. Kumar, "A performance analysis of DES and RSA cryptography," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 2(3), 2013, pp. 418-423.
- [10] S. Albermany, and F. Radihamade, "Survey: block cipher methods," *Int. J. Adv. Res. Technol*, vol. 5(11), 2016, pp. 11-22.
- [11] P. Sharma and R. Purohit, "Performance evaluation of symmetric block cipher RC6 with ECB and CBC operation modes," *International Conference on Intelligent Data Communication Technologies and Internet of Things, Coimbatore, India, 2018*, pp. 134-140.
- [12] W. Mazurczyk, "VoIP steganography and its detection—a survey," *ACM Computing Surveys (CSUR)*, vol. 46(2), 2013, pp. 1-21.
- [13] P. Asghari, A.M. Rahmani, and H.H.S. Javadi, "Internet of Things applications: A systematic review," *Computer Networks*, vol. 148, 2019, pp. 241-261.
- [14] S. Wachter, "Normative challenges of identification in the Internet of Things: Privacy, profiling, discrimination, and the GDPR," *Computer law & security review*, vol. 34(3), 2018, pp. 436-449.
- [15] M. A. Hossain, M. B. Hossain, M. S. Uddin, and S. M. Imtiaz, "Performance analysis of different cryptography algorithms," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6(3), 2016, pp. 659-665.
- [16] J.A. Dar, "Enhancing the data security of simple columnar transposition cipher by caesar cipher and rail fence cipher technique," *International Journal of Computer Science & Engineering Technology (IJCSSET)*, vol. 5(11), 2014, pp. 1054-1061.
- [17] Q. A. Kester, "A hybrid cryptosystem based on vigenere cipher and columnar transposition cipher," *International Journal of Advanced Technology and Engineering Research (IJATER)*, vol. 3(11), 2013, pp. 141-147.
- [18] G. Lasry, N. Kopal, and A. Wacker, "Cryptanalysis of columnar transposition cipher with long keys," *Cryptologia*, vol. 40(4), 2016, pp. 374-398.
- [19] S. R. Siregar, F. Fadlina, and S.D. Nasution, "Enhancing data security of columnar transposition cipher by fibonacci codes algorithm," *Third Workshop on Multidisciplinary and Its Applications, WMA-3, Medan, Indonesia, 2019*, pp. 1-10.
- [20] A. Bhowmic, and M. Geetha, "Enhancing resistance of hill cipher using columnar and Myszkowski transposition," *International Journal of Computer Sciences and Engineering*, vol. 3(2), 2015, pp. 20-25.
- [21] M. Sokouti, B. Sokouti, and S. Pashazadeh, "An approach in improving transposition cipher system," *Indian Journal of Science and Technology*, vol. 2(8), 2009, pp. 9-15.
- [22] F. Twum, J. Hayfron-Acquah, and W. Morgan-Darko, "A proposed enhanced transposition cipher algorithm based on rubik's cube transformations," *International Journal of Computer Applications*, vol. 182(35), 2019, pp. 18-26.
- [23] E. Surya, and C. Diviya, "A survey on symmetric key encryption algorithms," *International Journal of Computer Science & Communication Networks*, vol. 2(4), 2012, pp. 475-477.
- [24] P.C. Mandal, "Evaluation of performance of the symmetric key algorithms: des, 3des, aes and blowfish". *Journal of Global Research in Computer Science*, vol. 3(8), 2012, pp. 67-70.
- [25] E. A. M. Eliáš, "Evolutionary computation in cryptanalysis of classical ciphers," *Tatra Mt. Math. Publ.*, vol. 70, 2017, pp. 179-197.
- [26] S. Picsek, and D. Jakobovic, "Evolutionary computation and machine learning in cryptography," *Genetic and Evolutionary Computation Conference Companion, Cancún, Mexico, 2020*, pp. 1147-1173.
- [27] K.K. Pandey, V. Rangari, and S. Kumar, "An enhanced symmetric key cryptography algorithm to improve data security," *International Journal of Computer Applications*, vol. 74, 2013, pp. 0975 – 8887.
- [28] O. Omolara, A. Oludare, and S. Abdulahi, "Developing a modified Hybrid Caesar cipher and Vigenere cipher for secure data communication," *Computer Engineering and Intelligent Systems*, vol. 5(5), 2014.
- [29] P. Singh, and P. Shende, "Symmetric key cryptography: current trends", *International Journal of Computer Science and Mobile Computing*, vol. 3(2), 2014, pp. 410-415.
- [30] X. Yi, R. Paulet, and E. Bertino, "Homomorphic encryption," *Homomorphic Encryption and Applications*, 2014, pp. 27-46.
- [31] S. Srikanthaswamy, S. and D.H. Phaneendra, "Improved Caesar cipher with random number generation technique and multistage encryption," *International Journal on Cryptography and Information Security (IJCIS)*, vol. 2(4), 2012, pp. 39-49.
- [32] R. Rejani, and D.V. Krishnan, "Study of symmetric key cryptography algorithms," *International Journal of Computer Techniques*, vol. 2(2), 2015, pp. 45-50.
- [33] L. Keliher, "Refined analysis of bounds related to linear and differential cryptanalysis for the AES," *International Conference on Advanced Encryption Standard, Bonn, Germany, 2004*, pp. 42-57.
- [34] L. R. Knudsen, "Block Ciphers—a survey," *State of the art in applied cryptography*, Berlin, Heidelberg, 1998, pp. 18-48.
- [35] A. M. Kadhim, "Diagnosis of some cipher systems. journal of baghdad college of economic sciences university," vol. 4, 2013.