Word-based encryption algorithm using dictionary indexing with variable encryption key length

Ahmad Al-Jarrah[1], Amer Albsharat[2], Mohammad Al-Jarrah[2]
[1]Department of Applied Science Department, Ajloun University College, Al-Balqa Applied University, Salt, Jordan
[2]Department of Computer Engineering, Yarmouk University, Irbid, Jordan

Article Info

Article history:

Received Jan 5, 2021
Revised Jul 17, 2021
Accepted Aug 4, 2021

Keywords:

Encryption algorithm word-base
Encryption/decryption algorithm
English text encoding
Two-dimensional shuffling operations

ABSTRACT

This paper proposes a new algorithm for text encryption utilizing English words as a unit of encoding. The algorithm vanishes any feature that could be used to reveal the encrypted text through adopting variable code lengths for the English words, utilizing a variable-length encryption key, applying two-dimensional binary shuffling techniques at the bit level, and utilizing four binary logical operations with randomized shuffling inputs. English words that alphabetically sorted are divided into four lookup tables where each word has assigned an index. The strength of the proposed algorithm concluded from having two major components. Firstly, each lookup table utilizes different index sizes, and all index sizes are not multiples of bytes. Secondly, the shuffling operations are conducted on a two-dimensional binary matrix with variable length. Lastly, the parameters of the shuffling operation are randomized based on a randomly selected encryption key with varying size. Thus, the shuffling operations move adjacent bits away in a randomized fashion. Definitively, the proposed algorithm vanishes any signature or any statistical features of the original message. Moreover, the proposed algorithm reduces the size of the encrypted message as an additive advantage which is achieved through utilizing the smallest possible index size for each lookup table.

Corresponding Author:

Mohammad Al-Jarrah
Department of Computer Engineering, Yarmouk University
Irbid, Jordan
Email: jarrah@yu.edu.jo

1. INTRODUCTION

According to the enormous need for confidentiality in text-data communications, many researchers tackled and developed various techniques for data encryption such as advanced encryption standards (AES) [1], rivest-shamir-aldehman (RSA) [2], triple data encryption standard (TripleDES) [3], blowfish [4], twofish [5], where, each technique has its own merits [6], [7]. Nevertheless, most of these text-encryption techniques depend on having an encryption algorithm implemented on the character level through applying public or private keys [8]. Others used different encryption concepts on the binary level. This paper proposes an encryption technique applied on indices for english words obtained from a lookup table for English words instead of encrypting the words themselves.

In algorithms that use an encryption key technique, the encryption keys act as the parameters for mathematical formulas. For example, the well-known RSA algorithm, which is a deterministic one, uses public and private key encryption to encrypt a binary document [3], [9]. Many approaches extended RSA algorithm to encrypt text documents through representing text data in a binary manner [10]. Moreover, some algorithms utilize the concept of a shared dictionary which replaces some repeated characters with certain symbols such as the cheating text concept [11]. Al-Bsharat [12] developed an encryption algorithm for english language text. This technique restricted the english language that has no capital letters. The encryption technique is implemented on word level instead of letters where each English word is inserted into a dictionary.

This paper proposes an encryption algorithm that firstly encodes english words utilizing an index number to represent the word. Thus, the proposed technique produces a stream of binary code representation for the text document. In the second phase, the proposed algorithm converts the stream of binary code into a block of binary bits. This phase aims to vanish boundaries of word representations. In the last phase, the algorithm applies a set of logical shuffling operations in a randomized manner on the block of the binary code. The following sections discuss related work and the proposed algorithm in detail.

2. RELATED TEXT ENCRYPTION ALGORITHMS

Classical encryption can be divided into two categories namely substitution and shuffling. Substitution cipher methods of encryption utilize the idea of replacing each element of the text with another element. Shuffling algorithms rearrange the text elements in the document in a randomized fashion. Both categories utilized the character as the fundamental element. For example, RSA is known as an asymmetric cryptographic algorithm that would be used to encrypt text messages [3], [10], [13]. Kuryazov proposed a method to encrypt messages by points of elliptic curves using the optimal asymmetric data encryption algorithm [8]. Another related approach was playfair key matrix which is a three-dimensional representation of the Caesar' method [14], [15] leading to more complex decoding to achieve better confidentiality. Other techniques use the genetic algorithm for novel text encryption and decryption [9], [16], [17] using residue number system (RNS) by benefiting from inherent properties. Finally, a hybrid technique is proposed by Ramadan *et al.* [18] that uses volvox representation algorithm, which produces a positive integer number that helps in encryption/decryption of an english text.

Hwang [19] discussed the security flaws for the off-line micropayment scheme, which was proposed by Wuu *et al.* [20]. This off-line micro payment algorithm utilizes the asymmetric encryption scheme, which is used to provide electronic payments. The algorithm requires a small number of transactions where communication and computation costs should be very low. Al-Khateeb [21] proposed a methodology to encrypt and hide a text using deoxyribonucleic acid (DNA) coding. The encrypted text is hidden in an image where each letter is hidden in a pixel. Krishnan *et al.* [22] encrypts and decrypts text files using musical notes, where the user has to play music to have permission to open the file. The musical notes are used as a password to access the text file. The development of encryption/decryption methodology is not limited to software, but the hardware is also considered to enhance encryption/decryption techniques. For example, to enhance encryption algorithm performance, Zeebaree [23] proposes a hardware architecture to implement data encryption standards.

Other encryption techniques apply mechanisms such as the idea of pre-processing for encryption such as compression, which are categorized into two sub-classes. The techniques in the first sub-class depend on the implementation of a pre-processing process on the data elements (words), such as the cheating text technique [11], [24]. In the second sub-class, the file words are to be compressed in the pre-processing phase. Other pre-processing methods depend on working on the bit-stream of each byte (character) of the text files, such as the extensive bit-level encryption system (EBES) [25]. For example, bitwise randomization and serial bitwise feedback generation modules produce a new stream of randomized bits.

Intelligent dictionary based encoding (IDBE) [26] is another encoding strategy, which uses a text-based encoding algorithm for text data compression in order to have high-speed data transmission via the internet [27]. The strategy creates an intelligent dictionary at the first stage, which is used in the next step to encode the input text. First, the american standard code for information interchange (ASCII) code is used to encode the first 218 words in the input text by assigning each word one of the characters in the range 33-250. Then, the remaining words are encoded using two ASCII characters in the range 33-250, and so on.

Skibinski *et al.* [28] represent dictionary-based compression, where a word-replacing algorithm was´ implemented by relating each text word to a code word. Compression is achieved by choosing shorter-length code words. Mante suggests an encryption algorithm [29] based on a generated key. The key is generated using developed random matrices. A new algorithm by Alabdullah *et al.* [30] was proposed to use the reflection of binary search tree to minimize the overhead and achieve a high security level by using a dynamic offset. The advanced encryption algorithm (AES) is modified by Kareem *et al.* [31] to increase the security level achieved by the algorithm. The modification suggests replacing the XOR operation with (#) operation.

Other ideas come from combine two different natural languages to encrypt one by using the other language letters. Muhammad [32] proposed a new technique to encrypt english messages using kurdishe language letters and vice versa. Shareef *et al.* [33] encrypt Arabic text by converting Arabic letters firstly to latin letters. Then, the given text in Latin letters is encrypted/decrypted using the modified playfair cipher. Abusukhon *et al.* [34] add another security level based on the modification of text-to-image encryption (TTIE) algorithm by demonstrating the method of generating the encryption key.

3. WORD-BASED ENCRYPTION ALGORITHM (WBE)

Upon the previous literature review, many encryption methods have been successfully tackled text encrypting files through encoding the text at the character level [12], [35]. Due to the fact that the computational capabilities and memories of computers nowadays are enormous, this paper proposes word-based encryption algorithm (WBE) with two-dimensional shuffling operations. Firstly, the proposed algorithm encodes the message words with an index that is simply a binary number representing the word's location in a sorted dictionary. Thus, the text message is encoded into a binary stream. Next, the proposed algorithm converts the binary stream into a two-dimensional block of binary bits.

Moreover, the proposed algorithm utilizes an encryption key to largely randomize the two-dimensional binary block, which aims to vanish any statistical feature of the message and destroys any structural signature. This step includes a set of randomized bit-level logical two-dimensional shuffling operations. These operations assure the distribution of the bits of a word randomly. Thus, any group of bits in the output stream are indeed originated from different words. The following subsections discuss these three steps in detail.

3.1. Encoding text message utilizing word-based lookup tables

The proposed WBE encryption system suggests utilizing electronic dictionaries as lookup tables to encode text messages. Nowadays, electronic dictionaries for all spoken languages are available for researchers and exist for commercial uses. Furthermore, computational capabilities can perform extensive sorting/shifting of the dictionary elements within a very short period of time. In other words, the dictionary size will not degrade the performance of the proposed encryption algorithm especially if the process of the dictionary is applied once for every dictionary update. The proposed algorithm identifies language words as a unit of building messages. Thus, it utilizes a list of language words as a reference. The efficiency of the algorithm relies on the completeness of the list of language words. Nevertheless to say, no matter what the effort is paid to collect and add all words to the reference list, some messages may include words that are not found in the reference list. Moreover, some messages may use words originating from foreign languages, names for place, or symbols that will not be found in the word reference list. Therefore, the proposed algorithm should tackle all these issues through including a set of lookup tables to encode any message. The proposed algorithm suggests building four lookup table for the language words, letters, and symbols as follows:

3.1.1. The first lookup table (FiLT)

This table contains small and capital letters, the most used punctuation symbols (comma, dot, single quote, and double quote), and the space character. This lookup table has 64 characters and symbols. Figure 1 shows samples from FiLT table that includes English letters as shown in Figure 1(a), punctuation symbols as shown in Figure 1(b), and some control characters as shown in Figure 1(c). Therefore, each entry needs a 6 bits code. A seventh bit is added to indicate that the code was originated from the First Lookup table. Consequently, we used seven bits to encode this lookup table where the first bit is 0 and the other six bits for the entry code. This table is utilized to encode standalone letters, symbols, and words that are not found in the words' lookup tables described in the following items. The proposed algorithm suggests encoding the words that are not found in the words lookup tables by encoding them letter-by-letter as illustrated in the next section.

Letters	Code
A	0000000
B	0000001
Z	0011001
a	0011010
z	0110011

(a)

Symbols	Code
, (comma)	0110111
. (dot)	0111000
‘ (single quote)	0111001
“ (double quote)	0111010
(space)	0111011

(b)

Command	Code
Del	0111100
Capital	0111101
Start	0111110
New	0111111

(c)

Figure 1. The FiLT which includes: (a) English letters, (b) the most used punctuation symbols, (c) the commands

3.1.2. The second lookup table (SeLT)

This lookup table consists of numeric digits, all used symbols in English language and most frequent used words (an, the, and, at, he, she, but, from, to, be, of, in, that, have, it, for, not, on, with, as, you, do, by, this). We used eight bits to encode this table. The first two bits (10) are used to identify that the code is originated from the SeLT table. The next six bits are used to encode the entries in this table. The total available slots in this lookup table are 64, where 38 of them are used to encode the symbols and digits. Thus, 26 spaces are available for the most frequent words. According to statistical analysis accomplished in ref, we added the most frequent used words to this lookup table. Table 1 shows an example code for digits, symbols, and most frequent words.

Table 1. The second lookup table (SeLT) that includes numeric digits, symbols, and most frequents words

Word/Symbol/Digit	Code
0	10001011
and	10101000
the	10100111
#	10000001
$	10000010

3.1.3 The third lookup table (ThLT)

The words with two, three, four, and five characters were collected in this lookup table. Table 2 depicts samples from the third lookup table (ThlT). The number of entries in this table are 15297 words. Therefore, we need 14 bits code to encode all words (214=16 kilowords). To identify this lookup table, the code '110' is added at the beginning of each code. Thus, the size of code utilized for ThLT is 17 bits.

Table 2. The third lookup table (ThLT) that includes two, three, and four characters words

Word	Code
me	11000100101101101
car	11010101010010001
case	11010101010111011
about	11000000000100111
adopt	11000000001110100

3.1.4. The fourth lookup table (FoLT)

All remaining words in the dictionary are encoded in this lookup table. This table utilizes 20 bits to encode a word, where three bits (111) are used to identify this lookup table. Thus, the remaining bits (17 bits) are used to encode the words. Therefore, this table is capable of encoding 217 words. Table 3 presents code samples from FoLT table.

The proposed algorithm utilizes the four lookup tables to encode an English text where the word is the unit of encoding. Table 4 represents the lookup tables with encode size and its capacity. In the case where the word is not found in the SeLT, ThLT, or FoLT Lookup tables, the algorithm encodes it character by character utilizing FiLT table.

Table 3. The fourth lookup table (FoLT) that contains long words

Word	Code
abacas	11100000000000000111
abeles	11100000000001001111
abessive	11100000000001011010
abetments	11100000000001011100
abearing	11100000000001001100

Table 4. Proposed lookup tables; code length, number of entries (size), code prefix, and code range

Table	Code Length	Size	Code Prefix	Code Range
FiLT	7	2^6=64	0	0000000-0111111
SeLT	8	2^6=64	10	10000000-10111111
ThLT	17	2^{14}=16384	110	11000000000000000-11011111111111111
FoLT	20	2^{17}=131072	111	11100000000000000000-11111111111111111111

3.2. Text message encoding

The proposed WBE Cryptosystem starts by reading the text from its source word by word and build a corresponding stream binary code (SBC). Algorithm 1 shows text read steps from the input stream. Each time a word appears, the algorithm calls “Add” to add the binary code of the word to the SBC. Algorithm 2 illustrates the function “add,” which looks in the dictionary tables to find the word and returns its index. The word could be a standalone letter, symbol, or English word.

The proposed WBE algorithm assumes that the input text is written according to the formal written English language. Therefore, it assumes that each word or symbol is followed by space. Thus, space is used to separate words and symbols. Furthermore, the algorithm takes care of the letter case, where the small letter is the default case for all words. But the input text can have letters with upper and lower cases. Therefore, the algorithm should manipulate and take care of these cases. Thus, WBE algorithm encodes the word with upper and lower case and other special cases as follows:

3.2.1. Space case

The default case is that each word is followed by space. The algorithm adds the binary code for the word only (without the code for space). For example, if the input text is “he” (“he” followed by space), the algorithm adds the code for “he” only, which will be 14 bits code. For another example, the word he is followed by a comma, which is a punctuation symbol as in the part of the text (“he, who is known as ....”). In this case WBE algorithm adds the command code (“0111100”) after the binary code of the word “h” to inform the decoder not to add space. In this example (he, who is ...) the word “he” is encoded by 14 bits, then followed by the command word (“0111100”).

3.2.2. Unknown word

The text may have a word that is not in the dictionary (name for a person, places, medicine, cities). In this case, the proposed algorithm will encode this word letter by letter. To indicate that the inserted code is coded letter by letter, the code starts and ends with the following command code “0111111” The word is encoded letter by letter using FiLT table.

3.2.3. Word with upper and lower case letters

The proposed algorithm distinguish between upper and lower case letters. Each word can be one of the four cases; all letters are small, all letters are capital, the word starts with a capital letter, and some letters in the middle or the end are capital letters. In the first case, which is the default, the word consists of small letters. The algorithm adds the binary code for the word as it is coded in the dictionary. In the second case, where all letters of the word are capital letters, the algorithm adds the code “0111101” followed by the binary code for the word as if it is a lower case word. In the third case, where the word starts with a capital letter, the algorithm adds the code “0111110” followed by the code for the word as if it is a lower case word. For the last case where some characters in the middle or the end of the word are capital letters, the word is encoded as an unknown word as described earlier.

Algorithm 1. The algorithm for reading the text and build the corresponding binary code
```
Input: text: the input text from the file
Output: code: the corresponding binary code
begin {main}
isSymbol=false; // the variable is used to distinguish between characters and symbols
isWord = false; // the variable is used to make sure the collected token is word or not
word=“”; // initialize the word string to be empty
While !EOT Do // while not end of text
ch=text.getNextCh(); //(0, j) is the index of top-left cell in the diagonal
If ch is a letter then // read a character from a text
isWord=true;
word.append(ch);
else
if isWord then
isWord=false;
Add(word);
if ch !=space then
code.append (“0111100”);
Sym=ch;
isSymbol=true;
end
else
if ch=space then
if isSymbol then
Add(Sym);
isSymbol=false;
```

```
else
Add(Sym);
end
else
if isSymbol then
Add(Sym);
Code.append("0111100");
else
isSymbol=true;
end
Sym=ch;
end
end
end
end
end {main}
```

3.3. Preparing the stream of binary code (SBC) for encryption

In this step, the stream of encoded text is converted to a two-dimensional matrix of bits. For perfect shuffling and randomization, a square array matrix is recommended. Unfortunately, converting encoded text, which is a stream of bits, to a square matrix will not always be straightforward and not one-to-one conversions. To accomplish this, we performed the following:

a. In this step, we identify the order of the square of the matrix (N) as follow:

$$N = \lceil \sqrt{B} \rceil$$

where b is the size of the encoded stream in bits.

b. In this step, we create a square matrix of order N. The implementation is accomplished by creating a two-dimensional array with a size of (N by N). Then, we filled this matrix row by row until completing all encoded bits. The remaining elements are filled by zero.

Algorithm 2. Add(T): the add function encodes text according to the lock-up tables

```
Input: T: the input token to find the corresponding code for it
Output: code: the corresponding binary code.
begin {main}
if T in LUT then // looking for token T in the lookup table
FN=getFileNumber(); // return the file number which is one of the four look-up tables
N=LUT.indexOf(T.toLower()); // return the index of the token in the look-up table
If T.length()>1 Then
if T.isUpper() then
code.append("0111101");
else
if T.isStartUpper() then
code.append("0111110");
end
end
switch FN do
case 1 do
code.append("0"+N.toBinary(6));
Break;
end
case 2 do
code.append("10"+N.toBinary(6));
Break;
end
case 3 do
code.append("110"+N.toBinary(14));
Break;
end
case 4 do
code.append("111"+N.toBinary(17));
Break;
end
end
end
else
if T.length() > 1 then
code.append("0111111");
Add by chart(T);
code.append("0111111");
```

```
else
Output.error("Invalid character");
end
end
end {main}
```

The proposed conversion algorithm described above added a great value to the encryption process due to that each encoded word in the original text may be divided into one, two, three, or four bytes by remembering that the length of encoding size is not a multiple of eight. More precisely, any single byte will be part of one word or parts of two words. This means that shuffling and randomization described in the following section will not keep bits of any word adjacent.

3.4. Shuffling and randomization

An excellent encryption algorithm removes any statistical features such as signatures of frequent words that would lead to uncover the hidden message. The proposed algorithm would completely vanish any statistical features due to having a variable length word encoding approach. Moreover, the proposed code lengths are not multiples of 8 bits, meaning that our generated codes for words cannot be combined in a standard word size, where we utilized seven bits, seventeen bits, and twenty bits. Also, after we had encoded the message word, we mapped the resulted stream of bits to a two-dimensional matrix of bits which means that any codeword could be spanned on one, two, or three bytes. The next proposed step, which is Shuffling and Randomization, should shuffle adjacent bits randomly and apart in a non-detectable way. To achieve this, we first utilized a randomly generated encryption key that defines the method of shuffling and randomization. For shuffling and randomization, we defined four logical operations which will be applied on the square matrix to move bits of the encoded message away from each other. The following subsection describes these operations and the utilization of the encryption key.

3.4.1. Logical operations

For shuffling or swapping, four different functions had been defined. Each function implements rotating or circulation processes in a specific direction. These functions can be described as:

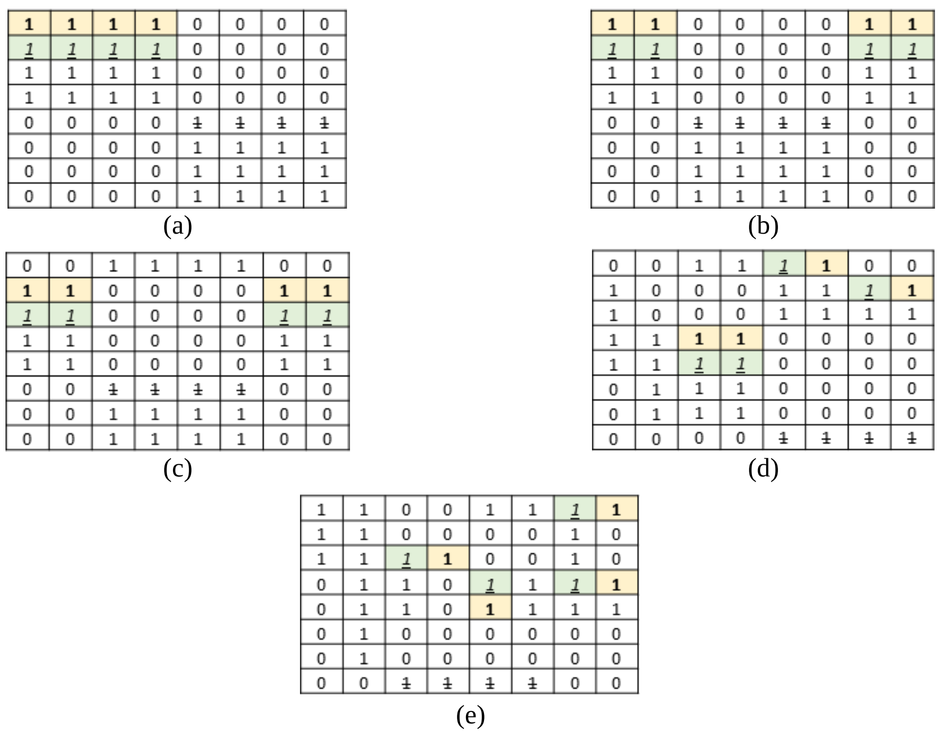
a. Horizontal rotation: This function, which is applied on the bit level, is a horizontal circulation of the square matrix. It takes two inputs which are the direction and number of bits to be circulated. The direction could be left or right. The number of shifts bits could be up to 128. The circulation is applied to each row of the square matrix. The result of horizontal rotation to the left direction by two steps for the matrix shown in Figure 2(a) is depicted in Figure 2(b).
b. Vertical rotation: The second function rotates the columns of the square matrix. The rotation cloud be up or down, and the number of rotation bits are up to 128 bits. Figure 2(c) shows an example of rotating the entire matrix one bit down.
c. Diagonal rotation: The third function is diagonal rotation. This operation rotates bytes in a diagonal direction. It has two inputs; the first input for the direction and the second one for the number of rotated bits. The direction could be right up or left down. Figure 2(d) shows one-byte left down diagonal rotation for a four by our matrix.
d. Circulation: This function forms a set of cycles and rotates the bits in the formed cycles clockwise or counter clockwise. The first cycle is formed by the first row, the last column, the last row, and the first column. The next cycle is formed by considering the next set o rows and columns adjacent to the first cycle. Figure 2(e) shows a circulation of two cycles clockwise for 8 by 8 matrix.

3.4.2. Encryption key

The encryption key, which is utilized as a symmetric encryption algorithm, ensures that the encrypted data can only be covered upon the availability of this key. Thus, the strength of the encryption algorithm relies on an undetectable key. Undetectable and unpredictable key's features are ensured through significant and variable size, which is randomly generated. The proposed algorithm utilizes a key with a set of binary bytes. The number of bytes is variable with at least fourteen bytes. Each digit of the key is considered a random input to one proposed logical operation to shuffle and randomize the bits of the encoded message shown in Algorithm 3.

3.5. The proposed encryption algorithm

The strength of the proposed encryption mechanism depends on randomizing the shuffling operations to vanish the expectation operation order. Moreover, the shuffling operations aim to remove any statistical or logical features. The proposed algorithm achieved these two properties via randomizing the shuffling process by randomizing the selection of the encryption key and removing the statistical and logical feature by applying the shuffling process on a bit level and imposing a variable number of shuffling repetitions with a minimum number of shuffling operations. The proposed algorithm utilizes the four logical shuffling functions discussed in subsection 3.4.1. The shuffling functions are applied according to the following order: i) function 0: horizontal rotations, ii) function 1: Vertical rotations, iii) function 2: Diagonal rotations, and iv) function 3: Circulation.



(a)

1	**1**	**1**	**1**	0	0	0	0
1	*1*	*1*	*1*	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
0	0	0	0	~~1~~	~~1~~	~~1~~	~~1~~
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

(b)

1	**1**	0	0	0	0	**1**	**1**
1	*1*	0	0	0	0	*1*	*1*
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
0	0	~~1~~	~~1~~	~~1~~	~~1~~	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0

(c)

0	0	1	1	1	1	0	0
1	**1**	0	0	0	0	**1**	**1**
1	*1*	0	0	0	0	*1*	*1*
1	1	0	0	0	0	1	1
1	1	0	0	0	0	1	1
0	0	~~1~~	~~1~~	~~1~~	~~1~~	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0

(d)

0	0	1	1	*1*	**1**	0	0
1	0	0	0	1	1	*1*	**1**
1	0	0	0	1	1	1	1
1	1	**1**	**1**	0	0	0	0
1	1	*1*	*1*	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	~~1~~	~~1~~	~~1~~	~~1~~

(e)

1	1	0	0	1	1	*1*	**1**
1	1	0	0	0	0	1	0
1	1	*1*	**1**	0	0	1	0
0	1	1	0	*1*	1	*1*	**1**
0	1	1	0	**1**	1	1	1
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	~~1~~	~~1~~	~~1~~	~~1~~	0	0

Figure 2. The four logical operations with an example: (a) the original matrix, (b) horizontal rotation to the left by two steps, (c) vertical rotation down by one step, (d) diagonal rotation down by two steps, (e) circulate counterclockwise by two steps

The proposed algorithm used a variable size binary encryption key with a minimum size equal to 14 bytes. The encryption key could be formed as:

$$Key = B_0B_1B_3 \dots B_n;$$

where, B means byte and n is the nth byte. The first byte of the Key $(B_0 = B_7B_6B_5B_4B_3B_2B_1B_0)$ initializes the main two parameters for the proposed algorithm. The value of bits (B_7B_{46}) of B_0 determines the number of the first shuffling function to start with. For example, if the value of (B_7B_{46}) equals (10)2, then the first shuffling operation will be Function 2, a diagonal circulation. The value of the remaining six bits multiplied by 10 is assigned to the shuffling repetition loop. If this value is less than 160, the shuffling parameter will be 160. Then, the next byte in the key will be the input to the running shuffling function. The shuffling functions take two inputs: the shuffling direction and the shuffling value. The first bit in the Key byte (B_7) will define the shuffling value. Algorithm 3 depicts the pseudo-code for the proposed algorithm.

Algorithm 3. Encryption key mapping with encryption operations
```
Input: CM: a square binary matrix of the encoded text. KEY: encryption key
Output: EncCM: CM after shuffling and randomization according to WCA algorithm
begin {main}
cyclicKey=Circular array(KEY); // create a cyclic array of size key
B=cyclicKey.getByte; // read first byte and point to next one.
ff=integer($B_7B_{46}$); // integer value for bit 7 and 6 of B.
rep=10*Integer($B_5B_4B_3B_2B_1B_0$); // integer value for bit 5 to bit 0 of B
Set count=0;
while count<max(160, rep) do
B=cyclicKey.getByte(); // read current byte and point to next one.
```

```
switch ff do
case 0 do
Horizontal Rotation(CM, B);
break;
end
case 1 do
Vertical Rotation(CM, B);
break;
end
case 2 do
Diagonal Rotation(CM, B);
break;
end
case 3 do
Circular Rotation(CM, B);
end
end
ff=(ff+1) mod 4;
end
end {main}
```

4. IMPLEMENTATION AND DISCUSSION

The proposed WBE algorithm has been implemented using C# language for the purpose of validation and evaluation. Figure 3 shows the program interface for WBE Windows application for the implementation of WBE algorithm. The main interface contains a menu as shown in Figure 3 that allows the user to use all algorithm features and functionality. The interface enables users to do the following:

a. File menu options as shown in Figure 4(a): the file options contain the main operation that the user needs:
   – New text as shown in Figure 4(a): the user can clear the text box to enter a new text.
   – Generate random text as shown in Figure 4(a): this option was used for the testing process, which generates a random text from the existing words in the dictionary.
   – Save file as shown in Figure 4(a): it allows the user to save the text opened or edited by the user to a standard format or UTF8 format.
   – Open text file as shown in Figure 4(a): the user can open the text file, which is mainly English text, to be encrypted via a provided operation on the main interface. The Interface has a text box that shows the opened text or allows the user to enter the text directly.

b. Operation menu options as shown in Figure 4(b): the operation menu contains the encryption and decryption options:
   – Encrypt as shown in Figure 4(b): the opened text or entered text to the text box can be encrypted using the Encrypt option in the operation option. The encrypted text will be saved in a file named by the user. The extension of the file will be ".wbe."
   – Decrypt as shown in Figure 4(b): the program allows the user to open the encrypted text from a file with the extension ".wbe" and run the decryption technique according to the entered key.

c. Tools menu options as shown in Figure 4(c): the tools menu allows the user to add and remove items to the dictionary:
   – Add to the dictionary: the provided dictionary files contain the most English language components (letters, symbols, and words). If the users want to add a new word or symbol, the program provides them with the required options to add it to the specific file.
   – Remove from the dictionary as shown in Figure 4(c): the user can remove a word that may be added incorrectly from the dictionary.

d. Statistics part as shown in Figure 4(d): the statistic part represents the statistical information about the input text; the number of characters, the text size based on ASCII code, the encrypted text size, the compression rate, and the number of words.

4.1. Results and discussion

Originally, WBE algorithm is designed to be a robust encryption algorithm that can decode an English text in a technique that prevents intruders and attackers from access the encrypted text. In addition to that, the algorithm compresses the text with a reasonable compression rate. In the following subsections, we analyzed the algorithm as an encryption algorithm and the compression rate for the encrypted file size compared with the original file size. For testing purposes, the algorithm used the following standard text files:

a. alice.txt: a well-known novel Alice in the wonderland by Carroll [36]. This is a novel written in the English language with an imaginary scenario.

b. plrabn12.txt: paradise lost is a poem for John Milton, who lived in the seventeenth century [37]. The file contains a well-known epic poem by John Milton, which is considered one of the best traditional novels. This poem provides a good spoken English language, having legendary expressions.
c. asyoulike.txt: the play "As You Like IT" by William Shakespeare is a comedy play written during the late years of the sixteenth century (perhaps 1599) and published during 1623 [38]. This comedy play provides a solid English language from the sixteenth century.
d. 2265.txt: the tragedies of hamlet are the longest play by William Shakespeare with 30,557 words [39].

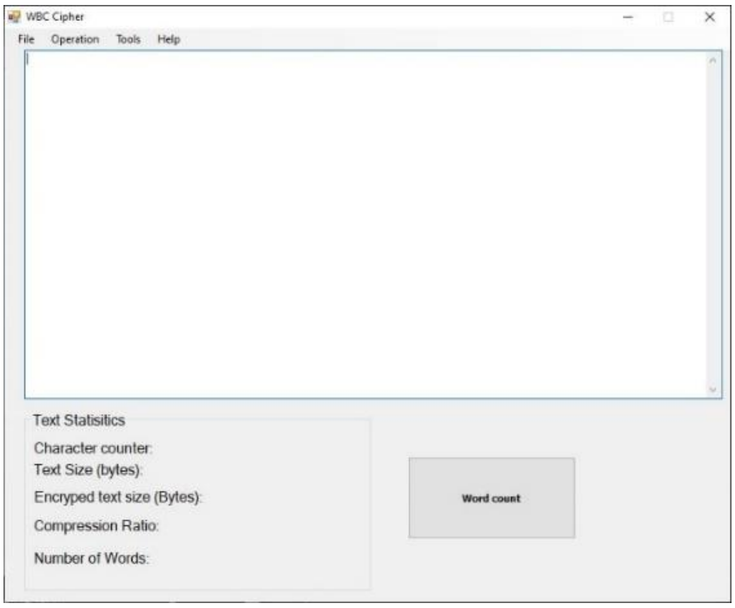


Figure 3. Main user interface of WBE implementation

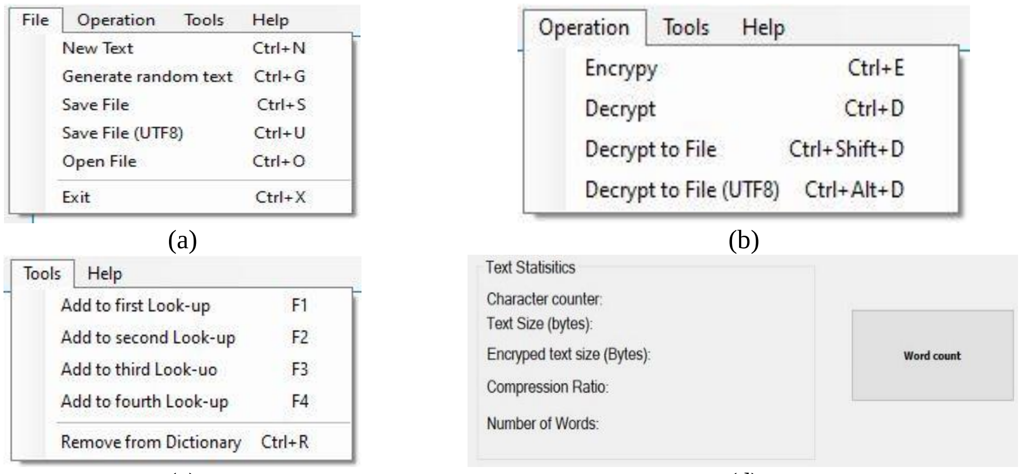


Figure 4. The four main parts of the application user interface: (a) file menu options, (b) operation menu options, (c) tools menu options, (d) text statistics

4.2. Strength of WBE encryption algorithm

The design of the WBE encryption algorithm aims to prohibit data access by unauthorized users. The encryption algorithm strength is evaluated by measuring the difficulties of breaking it by intruders attacks or analyzing its features that proves its immune against attacks [40]. Thus, the proposed encryption algorithm aims to make it very hard to follow or predict any signs or features that reveal the content. The power of the proposed WBE encryption algorithm can be analyzed using different types of known attacks, including “Bruteforce attacks,” “Codebook attacks,” “Known Plaintext attacks,” “Sniffers and Man in the Middle attacks,” and differential cryptanalysis.

4.2.1. Brute-force attacks

It is a decisive attack on cryptography, where the attacker tries all possible keys to get access to the encrypted data [41]. First of all, the proposed algorithm has variable encryption keys with a minimum length equal to fourteen digits. The bruit force trail would start with the minimum key length, which leads to $2^{14\times 8}=2.19\times 10^{33}$ possible key combinations. Then, it would try the next value of key length. This is a new philosophy of encryption key usage where the encryption user can randomly select the key’s size. Thus, for brute force attackers to extract the correct text, they have to try combining fourteen digits’ key, fifteen, and for all numbers above fourteen. The time required to run the algorithm and extract the hidden text may require hundreds of years. Moreover, the key can be changed via aging technique or data amount calculation technique, where the key must be changed after a certain period or after transmitting a certain amount of data.

4.2.2. Codebook attacks

Codebook attack benefits from having a specific block of text data always encrypted to the same block of encrypted data [42]. In other words, a particular word or sentence is always encrypted to the same encrypted binary value. However, the proposed WBE algorithm vanishes the codebook by randomizing the message code at the bit level. Thus, the shuffling process separates adjacent bits. This means that the set of bits for a word in the message will not be kept as one unit. For example, the bits that encode the word “the” in the first location in the message will be shuffled differently if this word appeared in another location in the message. In other words, if the word “the” is used multiple times in the message, the code for the “the” will be shuffled differently, and no signature may appear to indicate the existence of the word “the.”

4.2.3. Known plaintext attacks

The attack assumes having plain text, with the encrypted representation of that text, to do a particular reverse engineering process and get the encryption key [43]. The proposed WBE algorithm has two parameters that increase the complexity of applying this approach. The first one is the variable length key. The second repetition loop of the shuffling operations is variable and calculated from the encryption key. This means that the attacker cannot apply the reverse operations to perform the key generation because he must guess how many functions take place.

4.2.4. Sniffers and man in the middle attacks

Such attacks depend on having an attacker who listens to the traffic in the communication medium and tries to collect data to perform the proper analysis or get the key [44]. The proposed WBE algorithm applies different operations on the exact code representation based on its location in the encrypted. For example, let us assume that the word engineering” is used two times in the text. In the encoding stage, it will have the exact representation. But because the two representations in the two-dimensional binary matrix are in different locations, the shuffling operations applied in them are different. Thus, the distribution of the bits that belong to the first “engineering” word is entirely different from the distribution of the bits that belong to the representation of the second “engineering” word.

Man in the middle will receive a stream of binary bits. Consequently, the data collected by the man in the middle attacker cannot be analyzed via the assumption that they can know the representation of the encrypted data. Figure 2(a) shows a demonstration of the original two-dimensional matrix that represents the binary code message. The binary sequence ”1111” has been appeared twice. The first sequence is underlined, and the second one is strike-through. Figures 2(b), 2(c), 2(d), and 2(e) show the distribution of the first and second sequences after applying four shuffling operations. It is clear that the distributions of the two sequences are not correlated.

4.2.5. Differential cryptanalysis

This method tries to find the statistical feature for data elements in the encrypted data and compare it with the frequency of the spoken language elements, finding the frequency of the letters of the language or the most frequent words “strings of letters.” The proposed WBE algorithm is immune against such attack according to the following specifications:

a. Letter frequency: this algorithm depends on encryption on the word level, which makes it impossible to get the letters.
b. Word frequency: as we mentioned before, the words encoding length is not constant and is not multiple bytes. Moreover, the shuffling process is applied on the bit level and displaces adjacent bit away from each other randomly.

c. Probability of breaking the cryptosystem: the probability of breaking the proposed WBE cryptosystem depends on many factors including knowing the sequence of shuffling functions, what the shuffling functions do, and how many times the shuffling is applied. Moreover, the key length is variable and unknown to the attacker, which means that the attacker should try different key sizes.

Estimating the probability of guessing swapping functions for one loop for the proposed shuffling algorithm shown in algorithm 3 is equal to the probability of predicting the direction of shuffling times, the probability of predicting the number of shifts. According to the parameters used in the proposed algorithm as illustrated in Algorithm 3, we could define this probability as (1):

$$Ps = \frac{1}{2} \times \left(\frac{1}{2}\right)^7 = \left(\frac{1}{2}\right)^8 \tag{1}$$

where *Ps* represent the probability of predicting one key byte from the encryption key, which is used as a parameter for the operation inside the shuffling loop, then, the probability to predict all bytes of keys used for the shuffling functions is (2):

$$Pkey = (Ps)^{\max(10,rep)} \tag{2}$$

where *rep* is the integer value for the six bits of the first byte in the encryption key, which defines the value of the repetition loop of the proposed algorithm. If we consider the minimum value for the function max(10, rep), which is 10, the highest *Pkey* becomes:

$$Pkey = (Ps)10$$

Hence, according to the proposed WBE algorithm, the prediction of the first shuffling function is equal to 1/4. Thus, the overall probability prediction of the key becomes:

$$\text{PTbkey} = \frac{1}{4} \times \frac{1}{rep} \times (Ps)^{10} = \left(\frac{1}{2}\right)^8 \times \left(\left(\frac{1}{2}\right)^8\right)^{10} = \left(\frac{1}{2}\right)^{88} = \frac{1}{(3.094 \times 10^{26})} \tag{3}$$

Then, the number of attempts for decrypting the encrypted file with the assumption that the attacker knows the key length and the exact process done by each of the shuffling functions would be: 3.09×10^{26} attempts. Let us assume that each attempt requires 1 *ms*. Then, the time to break this algorithm is (4):

$$BT = 3.09 \times 10^{26} \times 10^{-3} seconds = 9.7 \times 10^{26}\ years \tag{4}$$

If we considered the average value for repetition in the proposed algorithm, which is equal to 32 (*rep* = 32), the total number of needed attempts (with the assumption that the attacker knows the exact processes of the swapping functions) would be 2^{264}=2.964×10^{79}.

4.3. Compression

The size of the produced encrypted message by the proposed encryption algorithm is less than the original message. Thus, the proposed WBE algorithm compresses the text message as an additive feature. In most cases, modern encryption algorithms increase the size of encrypted with respect to the actual size of the data [45]. Such an issue increases the size of the utilized memory and the size of transmitted data. Thus, a compression process usually takes place before encrypting data, where the compression depends on having redundant data [28], [46]. Then, the compressed version of that data is encrypted.

In general, compression can be divided into two main categories; lossless and lossy. The compression of text should always use lossless ones, which means that the compressed data must lead to the same uncompressed data when extracted. One of the essential primary evaluations of the compression algorithm is the compression ratio. The compression ratio (CR) is defined as (5).

$$CR = \frac{Size\ of\ Compressed\ Data}{Data\ Size\ Before\ Compression} \tag{5}$$

The proposed WBE algorithm compresses the input text to a good compression ratio as an additive feature. Table 5 shows the compression results for the WBE algorithm compared with several known algorithms. The compression ratio of the proposed WBE was tremendously efficient.

Table 5. Comparison between the proposed WBE compression ratio and compression ratio for known algorithms

Test File	Original File Size (Bytes)	ZIP Comp. File Size	ZIP Comp. Ratio	RAR Comp. File Size	RAR Comp. Ratio	WBE Comp. File Size	WEP Comp. Ratio
Alice29	152087	55854	0.367	51293	0.337	88529	0.582
Asyoulike	162250	59338	0.365	55100	0.339	98878	0.609
2265	184403	74998	0.406	70921	0.384	130304	0.707
Plrabn12	493523	204564	0.414	180761	0.366	258062	0.522

5. CONCLUSION

This paper proposes a novel English text encryption algorithm WBE, that is immune against known attacks. The proposed algorithm adopts the english word as a unit of encoding, utilizes a variable length encryption key, applies two-dimensional binary shuffling techniques, and utilizing four binary logical operations with randomized shuffling inputs. The proposed algorithm strength has been achieved due to many facts. First, the proposed algorithm utilizes variable-length indices to encode English words. Then, it utilizes multiple shuffling logical operations with variable inputs based on the randomized encryption key, and the shuffling operation is applied on a two-dimensional binary matrix. Hence, the goal of encoding the text on the word level and shuffling the encoded stream at the bit level is to separate adjacent bits away from each other in a random fashion. Finally, the proposed algorithm implementation and analysis showed its superiority against most types of attacks. For example, statistical analysis proved that breaking WBE might require thousands of years utilizing current computer computational power.

REFERENCES

[1] S. Heron, “Advanced encryption standard (AES),” *Network Security*, vol. 2009, no. 12, pp. 8-12, 2009, doi: 10.1016/S1353-4858(10)70006-4.

[2] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, “Key-privacy in public-key encryption,” in *International Conference on the Teory and Application of Cryptology and Information Security*, 2001, vol. 2248, pp. 566-582, doi: 10.1007/3-540-45682-1_33.

[3] S. Wang and G. Liu, “File encryption and decryption system based on RSA algorithm,” *2011 International Conference on Computational and Information Sciences*, 2011, pp. 797-800, doi: 10.1109/ICCIS.2011.150.

[4] S. Manku and K. Vasanth, “Blowfish encryption algorithm for information security,” *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 10, pp. 4717-4719, 2015.

[5] B. Schneier, “The twofish encryption algorithm,” *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, vol. 23, no. 12, pp. 30–34, 1998.

[6] M. A. Mushtaque, H. Dhiman, S. Hussain, and S. Maheshwari, “Evaluation of des, tdes, aes, blowfish and two fish encryption algorithm: based on space complexity,” *International Journal of Engineering Research and Technology (IJERT)*, vol. 3, no. 4, 2014.

[7] D. A. V. B. Devasish Pal, and P. Raghavendra, “An intelligent method of secure text data transmission through internet and its comparison using complexity of various indian languages in relation to data security,” *Global Journal of Computer Science and Technology*, vol. 13, no. 4, 2013.

[8] D. M. Kuryazov, “Optimal asymmetric data encryption algorithm,” *Global Journal of Computer Science and Technology*, vol. 21, no. 2-H, 2021. Accessed: Apr. 19, 2021. [Online]. Available: https://computerresearch.org/index.php/computer/article/view/2028

[9] M. Alruily, O. R. Shahin, H. Al-Mahdi, and A. I. Taloba, “Asymmetric DNA encryption and decryption technique for Arabic plaintext,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–17, 2021, doi: 10.1007/s12652-021-03108-w.

[10] P. M. Aiswarya, A. Raj, D. John, L. Martin, and G. Sreenu, “Binary RSA encryption algorithm,” *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2016, pp. 178-181, doi: 10.1109/ICCICCT.2016.7987940.

[11] C. Rupa, P. S. Avadhani, and V. Nagalakshmi, “Fast comparison encryption scheme using cheating text technique,” *International Journal of Engineering Science and Technology*, vol. 2, no. 6, pp. 1725–1728, 2010.

[12] A. M. Al-Bsharat, “Developing a word-based encryption algorithm,” M.S. Thesis, Hijjawi Faculty for Engineering and Technology, Department of Computer Engineering, Yarmouk University, Irbid, Jordan, 2016.

[13] R. Yu *et al.*, “Authentication with block-chain algorithm and text encryption protocol in calculation of social network,” in *IEEE Access*, vol. 5, pp. 24944-24951, 2017, doi: 10.1109/ACCESS.2017.2767285.

[14] S. A. Khan, “Design and analysis of playfair ciphers with different matrix sizes,” *International Journal of Computing and Network Technology*, vol. 3, no. 03, pp. 117-122, 2015, doi: 10.12785/ijcnt/030305.

[15] M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, and K. Satori, “A novel text encryption algorithm based on the two-square cipher and caesar cipher,” in *International Conference on Big Data, Cloud and Applications*, 2018, vol. 872, pp. 78-88, doi: 10.1007/978-3-319-96292-4_7.

[16] P. A. N. Agbedemnab, E. Y. Baagyere, and M. I. Daabo, “A novel text encryption and decryption scheme using the genetic algorithm and residual numbers,” in *Proceedings of 4th International Conference on the Internet, Cyber Security and Information Systems*, 2019, vol. 12, pp. 20-31, doi: 10.29007/zd9h.

[17] R. B. Abduljabbar, O. K. Hamid, and N. J. Alhyani, “Features of genetic algorithm for plain text encryption,” *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 434-441, 2021, doi: 10.11591/ijece.v11i1.pp434-441.

[18] S. A. M. Ramadhan, H. M. Yahya, and R. M. Alyas, “Design a hybrid technique based new genetic approach for text encryption,” *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 16, no. 1, 2018.

[19] S. J. Hwang, “Security flaws of off-line micro payment scheme with dual signatures,” in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 2014, vol. 260, pp. 905-909, doi: 10.1007/978-94-007-7262-5_103.

[20] L. C. Wuu, K. Y. Chen, and C. M. Lin, “Off-line micro payment scheme with dual signature,” *Journal of Computers*, vol. 19, no. 1, 2008.
[21] Z. N. Al-Khateeb and M. Jader, “Encryption and hiding text using DNA coding and hyperchaotic system,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 2, pp. 766-774, 2020, doi: 10.11591/ijeecs.v19.i2.pp766-774.
[22] R. Krishnan, R. Thomas, D. Akshay, V. S. Krishna, and R. Divya, “An intelligent text encryption system using musical notes,” *Innovative Data Communication Technologies and Application*, 2021, vol. 59, pp. 449-459, doi: 10.1007/978-981-15-9651-3_38.
[23] S. R M. Zeebaree, “DES encryption and decryption algorithm implementation based on FPGA,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 18, no. 2, pp. 774-781, 2020, doi: 10.11591/ijeecs.v18.i2.pp774-781.
[24] S. Porwal, Y. Chaudhary, J. Joshi, and M. Jain, “Compression methodologies for lossless data and comparison between algorithms,” *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 2, pp. 142-147, 2013.
[25] S. Roy, “The extensive bit-level encryption system (EBES),” *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 5, no. 5, pp. 67-73, 2013, doi: 10.5815/ijitcs.2013.05.09.
[26] V. Govindan and B. S. Mohan, “An intelligent text data encryption and compression for high speed and secure data transmission over internet,” in *International Conference on Information Technology Coding and Compression (ITCC)*, 2005.
[27] S. Shanmugasundaram and R. Lourdusamy, “Text preprocessing using enhanced intelligent dictionary based encoding (EIDBE),” *2011 3rd International Conference on Electronics Computer Technology*, 2011, pp. 451-455, doi: 10.1109/ICECTECH.2011.5941833.
[28] P. Skibinski, S. Grabowski, and S. Deorowicz, “Revisiting dictionary-based compression,” *Software: Practice and Experience*, vol. 35, no. 15, pp. 1455-1476, 2005, doi: 10.1002/spe.678.
[29] P. G. Mante, H. R. Oswal, D. Swain, and D. Deshpande, “A symmetrical encryption technique for text encryption using randomized matrix based key generation,” in *Advances in Data Science and Management*, 2020, pp. 137-148, doi: 10.1007/978-981-15-0978-0_13.
[30] B. Alabdullah, N. Beloff, and M. White, “E-ART: A new encryption algorithm based on the reflection of binary search tree,” *Cryptography*, vol. 5, no. 1, 2021, doi: 10.3390/cryptography5010004.
[31] S. M. Kareem and A. M. S. Rahma, “New method for improving add round key in the advanced encryption standard algorithm,” *Information Security Journal: A Global Perspective*, pp. 1-13, 2021, doi: 10.1080/19393555.2020.1859654.
[32] F. Rashid, “Design and implementation a new approach for enhancing encryption and decryption mechanisms,” *Clemson University – School of Computing*, 2020, doi: 10.2139/ssrn.com.3590807.
[33] I. R. Shareef, N. D. Al-Shakarchy, E. H. Abd, D. A. Al-Nasrawi, H. F. Al-Shahad, and H. J. Aleqabie, “New cryptographic system of romanized Arabic text based on modified playfiar,” *Journal of Engineering and Applied Sciences*, vol. 14, no. 4, pp. 1331-1338, 2019.
[34] A. Abusukhon, M. N. Anwar, Z. Mohammad, and B. Alghannam, “A hybrid network security algorithm based on diffie hellman and text-to-image encryption algorithm,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 1, pp. 65-81, 2019, doi: 10.1080/09720529.2019.1569821.
[35] D. Pal, P. Raghavendra, and A. V. Babu, “An intelligent method of secure text data transmission through internet and its comparison using complexity of various indian languages in relation to data security,” *Global Journal of Computer Science and Technology*, vol. 13, no. 4, 2013.
[36] L. Carroll, “Alice’s Adventures in wonderland,” *The Project Gutenberg EBook of Alice in Wonderland, 1st Ed*., 2006. [Online]. Available: http://archive.org/stream/alicesadventures19033gut/19033.txt
[37] J. Milton, “Paradise Lost,” *Gutenberg*, 1992, pp. 1608-1674. [Online]. Available: www.gutenberg.org/ebooks/26
[38] W. Shakespeare, “As You Like It,” *Project Gutenberg*, 1998. [Online]. Available: https://www.gutenberg.org/ebooks/1523
[39] W. Shakespeare, “The Tragedi of Hamlet,” *The Project Gutenberg's Etext of Shakespeare's First Folio*, 2000. [Online]. Available: http://www.gutenberg.org/files/2265/2265.txt
[40] T. Soe, S. S. Mon, and K. A. Thu, “Performance analysis of data encryption standard (DES),” *International Journal of Trend in Scientific Research and Development*, vol. 3, no. 5, pp. 1439-1443, 2019.
[41] K. Apostol, “Brute-force Attack,” Salu Press, 2012, doi: 10.1007/springerreference_9302.
[42] R. L. Rivest, “All-or-nothing encryption and the package transform,” *in International Workshop on Fast Software Encryption*, 1997, vol. 1267, pp. 210-218, doi: 10.1007/BFb0052348.
[43] U. Meyer and S. Wetzel, “A man-in-the-middle attack on umts,” in *Proceedings of the 3rd ACM Workshop on Wireless Security*, 2004, pp. 90-97, doi: 10.1145/1023646.1023662.
[44] E. Biham and A. Shamir, “Differential cryptanalysis of the data encryption standard,” *Science and Business Media*, 2012.
[45] W. Freeman and E. Miller, “An experimental analysis of cryptographic overhead in performance-critical systems,” *MASCOTS '99. Proceedings of the Seventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 1999, pp. 348-357, doi: 10.1109/MASCOT.1999.805073.
[46] S. Natarajan, M. Ganesan, and K. Ganesan, “A novel approach for data security enhancement using multi level encryption scheme,” *International Journal of Computer Science and Information Technologies*, vol. 2, no. 1, pp. 469-473, 2011.

BIOGRAPHIES OF AUTHORS

Ahmad Al-Jarrah is an assistant professor at the Computer Science Department, Al-Balqa Applied University, Jordan. Dr. Al-Jarrah holds Ph.D. in Computer Science from Computer Science Department, New Mexico State University, USA (2016). In addition, he received his B.Sc. in Computer Science at Irbid National University, Jordan, and Master degree from the Computer Science department at Yarmouk University, Jordan. His research interest spans the general areas of online learning, social media, collaborative learning, and software engineering. He can be contacted at email: aljarrah@bau.edu.jo.

Amer Albsharat is a cyber-security and cryptography expert. As a CEO of Smart Business Systems, he specializes in enterprise cyber security consulting and risk management services. With a solid experience in business continuity, disaster recovery, and information security management systems, Amer has a major focus on research and development in cryptography, artificial intelligence, and embedded systems. He can be contacted at email: absharat@gmail.com.

Mohammad Al-Jarrah is a professor of Computer Engineering at Yarmouk University. He earned his Ph.D. in 2000 from University of Ohio, USA., MS and BS in Computer Engineering from Jordan University of Science and Technology, Jordan in 1992, and 1990. Since 2000, he has been working with the Department of Computer Engineering at Yarmouk University. His research interests include image indexing and retrieval, multimedia systems, distributed systems, medical imaging, network management and security, data encryption, and many others. He can be contacted at email: jarrah@yu.edu.jo.