Dictionary Based Arabic Text Compression and Encryption Utilizing Two-Dimensional Random Binary Shuffling Operations

Ahmad Al-Jarrah
Applied Science Department, Al-Balqa Applied University, Jordan
aljarrah@bau.edu.jo

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

Amer Albsharat
Computer Engineering Department, Yarmouk University, Jordan
absharat@gmail.com

Abstract: *This paper developed Arabic text encryption and compression based on dictionary indexing algorithm. The proposed algorithm includes encoding the Arabic text utilizing Arabic words dictionary, mapping encoded binary stream to a two-dimensional binary matrix, utilizing randomized variable size encryption key, applying randomly binary shuffling functions on the two-dimensional matrix, and mapping back the two-dimensional binary matrix into a sequential binary stream. The decryption algorithm at the receiver side implements the encryption steps reversely, utilizing the encryption key and the shared Arabic word dictionary. In this dictionary, the words of the formal Arabic language are classified into four categories according to the word length and sorted alphabetically. Each dictionary category is given an index size that is large enough to fit all words in that category. The proposed algorithm shuffles adjacent bits away from each other in random fashion through utilizing randomized variable length encryption key, two-dimensional shuffling functions, and repetition loop. Moreover, the index size is selected not to be multiple bytes to destroy any statistical feature that may be utilized to break the algorithm. The proposed algorithm analysis concluded that it could be broken after $3.215 * 10^9$ years. Moreover, the proposed algorithm achieved a less than 30% compression ratio.*

Keywords: *Encryption algorithm, decryption algorithm, compression algorithm, arabic text encryption, arabic text decryption, arabic text encoding, two-dimensional binary shuffling.*

Received December 23, 2020; accepted January 9, 2022
https://doi.org/10.34028/iajit/19/6/3

1. Introduction

The enormous need to use cryptography had existed since 1900 B.C. when an Egyptian used non-standard hieroglyphs in an inscription [20]. The meaning of cryptography is a Greek word that consists of two parts: the first part means “hidden or secret,” and the second part means “description.” Encryption and decryption are the two opposite processes; where the first one is concerned with transforming information to make it secret and cannot be read by anyone other than those with the unique key. The decryption process is the reverse process of the encryption where it concerns with transforming the hidden encrypted information to the original form. Nowadays, people worldwide use the Internet to transmit data and messages. This communication causes a considerable number of text messages to be transmitted over the communication media daily. The enormous need to connect people worldwide has caused communication tools and technologies to flourish rapidly in the last decade.

Moreover, in addition to transferring information over communication media confidentially and securely, the need to decrease the size of the transferred data becomes a necessity. The high speed in the wheel of developing Internet technologies forces researchers to implement various data encryption and compression techniques. Each of these techniques has its advantages and disadvantages. The text-encryption techniques have been implemented in different text code levels, such as character level, binary level, or word level. Most popular techniques are implemented on the character level.

Confidentiality, integrity, and availability are the main three concerns of any information security system [4, 11, 20, 30]. The secret information should be accessible within a specific time by authorized users, where confidentiality guarantee that unauthorized users couldn’t access the encrypted information. The encryption algorithm should guarantee that the secret/hidden information is kept confidential and cannot be accessed by an attacker. In addition to the three concerns, to secure the transferred messages, the encryption algorithm should consider the size of the encrypted message and the time complexity for encryption and decryption algorithms.

In algorithms that use encryption key techniques, the encryption key act as the parameters for the formulas of these algorithms. For example, the well-known RSA algorithm uses public-key encryption with an intense amount of traffic needed to transfer these keys, especially when having high exponential factors since this algorithm is a deterministic one. A number of the new approaches use the idea of representing text data in a binary manner, then encrypting them via Rivest-Shamir Adlmeman (RSA) [32].

In the last decade, many researchers have tackled Arabic text encryption. Algahtani *et al.* [8] developed a new way of Arabic encryption using Arabic plaintext by modifying The Vigenère cipher by repeatedly adding a key into the plaintext. Others discuss algorithms using the concept of a shared dictionary that depends on replacing some repeated characters with certain symbols for the purpose of compression such as the cheating text concept [19]. Other types of algorithms use the idea of a common dictionary. Al-Bsharat [3] introduced a new cryptosystem that encrypts English language text. Al-Bsharat technique restricts that it works only for English with no capital letters. The encryption technique is implemented on word level instead of letters where each English word is inserted into look-up tables. Al-Jarrah *et al.* [5] developed a new technique to work to let Al-Basharat technique works on all cases.

In this paper, we proposed and implemented a new compression and encryption algorithm for Arabic text where Arabic words are the base of encoding. Each Arabic word has been given a unique code by storing Arabic words in dictionary tables. Firstly, the Arabic message is encoded utilizing a dictionary table into a binary stream. Then, the generated binary stream is mapped to the two-dimensional binary matrix. After that, the proposed algorithm randomly shuffles the two-dimensional binary matrix by applying two-dimensional shuffling operations. The parameters for the shuffling operations are generated from the encryption key. The proposed encryption algorithm utilizes a variable-length encryption key. The randomly generated encryption key carries all inputs for the proposed encryption algorithm, such as the number of repetitions of shuffling operations, the next shuffling operation to be applied, and the input of the shuffling operations, which will be discussed in detail in section 2.5. The proposed algorithm, which imposes randomness in all steps, improves the security level if compared with the character-based encryption algorithm.

1.1. Literature Review

The encryption algorithms are classified into two primary techniques. The first one depends on replacing the text code with another one, and the second technique depends on shuffling the exact text by rearranging the text code. Different algorithms that implement both techniques were implemented on character-level. Wang and Liu [31] invited the permutation method that leads to shuffling characters' places. It shifts each character three positions in the alphabetic order. The other approach based on 3-dimensional representation of Caeser's method [13] leads to more confidentiality and more complex decoding.

Some research applied pre-processing steps before encryption, such as compression. These techniques, which apply bitwise randomization and serial bitwise feedback generation modules, produce a new stream of randomized bits. The Extensive Bit-level Encryption System (EBES) is an example of a technique that depends on working on the bit-stream of each byte (character) of the text files [16]. Implementing these techniques provides an efficient algorithm that can be used to transfer short text over media such as passwords and critical short-length texts.

Green discussed the off-line micropayment scheme [14]. The algorithm is used to support electronic payments by utilizing an asymmetric encryption scheme. The main assumption of the algorithm is that the amount of payments are small. Therefore, the algorithm aims to reduce the communication cost. To have high-speed data transmission over the Internet, an intelligent method of secure text utilizes a text-based encoding algorithm for text compression [24]. A transformation process is implemented on text elements to generate lightweight words that look similar to the processed words. This algorithm depends on the idea that our visual recognition of the two words is analogous. Also, Roy [26] 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.

Furthermore, an intelligent approach utilizes a pairing function to distribute one message to two different messages [24]. This method depends on encrypting the single message into two messages with the help of pairing functions. On the receiver side, two messages will be decoded into single messages with the help of the pairing function.

Arabic language text was an objective of many researchers [21, 29] to build encryption/decryption such as algorithms developed by Alqahtani, Al-Omary, and Kuppuswamy [6, 8, 20]. Alqahtani *et al.* [8] algorithm depends on adding a key repeatedly into the plain text where this approach is developed based on the idea of Vigenere cipher. Furthermore, Al-Omary [6] presents another technique that depends on using a shared key. This algorithm was implemented in the old order, which is called ABJAD order of Arabic letters instead of the normal alphabetical order. Moreover, Kuppuswamy and Alqahtani [20] studied the effectiveness of a symmetric-key algorithm on Arabic characters. The algorithm is based on assigning integer values to the Arabic letters and digits, which yields simple and effective encryption and decryption procedures. Karima *et al.* [17] studied Arabic text to develop good text categorization. The study help understand the Arabic text deeply, leading to building better encryption algorithms.

Aysan and Kuppuswamy [11] proposed a hybrid encryption algorithm that is based on simple multiplication and logarithm function with Caesar cipher. The algorithm takes care of the required time and memory space to implement the algorithm in addition to having a secured encrypted text. Another technique presented by Kataria *et al*. [18] is based on the steganography approach. The ExOr operation is used in the steganography approach where the operation is executed on two characters, leading to reordering them to have more secured results output.

Alsuhibany [9] introduced the Visual Cryptography (VC) algorithm as a new version of cryptography. It is a new approach that doesn't need an encryption or decryption key. Alsuhibany [9] developed a tool that uses VC approach to encrypt/decrypt images of Arabic text. Furthermore, a mathematical equation was used by Shaban [27] to introduce a new encryption algorithm for Arabic text. The developed algorithm is efficient in terms of the difficulty of guessing the encryption key and hence the encrypted text. Hence more, Shareef *et al*. [28] developed a cryptographic system that consists of three steps; romanize Arabic text, encrypt/decrypt the Romanized text using Playfair cipher, a Knight tour key is generated to be used in the encryption/decryption algorithm. The third step is generating Arabic text by deromanizing process.

On the other side of research, researchers studied a Neural network beside a genetic algorithm to be used in Arabic cryptography techniques [1, 7]. Rihan and Osma [25] conducted an experiment to evaluate a cryptography technique using a Neural network for the Arabic language. Abduljabbar [2] developed a new approach to encrypt Arabic text using a genetic algorithm. The algorithm generates a chromosome with 8 bits size by selecting a crossover point randomly. The results showed better performance with respect to other encryption techniques such as RSA, Data Encryption Standard (DES), etc., Habeeb [15] used a genetic algorithm to attack encrypted Arabic text. Habeeb [15] applied the attack on different text sizes with different encryption key lengths. The result showed that the algorithm could decrypt 100% of the ciphertext with a size in the range of 600 to 1000 letters that have been encrypted using five letters encryption key.

The remainder of this paper is organized as follows, section 2 explains the details of the proposed algorithm. Section 3 presents the compression results as an output of proposed algorithm. In section 4, the implementation and the results of the algorithm are discussed with the details of created software. The conclusion is presented in section 5 with the future works.

2. The Algorithm

The proposed algorithm for compression and encryption of Arabic text combines compression and encryption together in one algorithm, emphasizing an innovative and very immune encryption technique. This algorithm has been designed and optimized to encrypt and compress Arabic text. In addition to reducing the size, the compression part has increased encryption complexity. Moreover, the algorithm assumes that the recipient of the encrypted text has the complete dictionary that has been used encryption process. Figure 1 represents the general steps for compressing and encrypting Arabic text. Firstly, the proposed algorithm encodes Arabic text to a binary stream utilizing Arabic word dictionary tables indices. The dictionary tables consist of all known Arabic words. The size of the dictionary tables that we used contains 9,196,323 words. Moreover, the Arabic word lengths range from two letters to more than 13 letters (a few words have one letter length; the system manipulates these words as letters).

Table 1 categorize Arabic words according to their length, where the dictionary tables are prepared by dividing the words into four groups according to the words' length. The algorithm assigns each dictionary table a specific coding pattern. Then, the obtained binary stream is mapped into the two-dimensional binary matrix. Finally, the binary matrix is shuffled utilizing two-dimensional shuffling logical operations randomly. The shuffling operations are controlled by a randomly generated encryption key. The generated encryption key has a variable length to make the encryption algorithm more powerful and unbreakable. The following subsection describes the proposed algorithm in detail.

Table 1. Arabic words statistics and categorization based on the word length.

Category	**Word length (character)**	**# of words**	**Total of Categories**
Characters	1	60	60
Short words	2	441	10617
	3	10176	
Medium words	4	95171	1587483
	5	420928	
	6	1071384	
Long words	7	1904365	7598223
	8	2299474	
	9	1887445	
	10	1009748	
	>10	497191	
Total		9196323	9196323

2.1. Preparing Arabic words Dictionary Tables

Before working on the proposed compression and encryption process of the Arabic text, we need to prepare a set of encoding dictionaries for Arabic words. We started with the Arabic word list, which was prepared by Attia [10], which includes most of the Arabic words. To ensure that we have all Arabic words in the encoding dictionaries, the list was compared with the Holly Quran words [12]. The missed words are added to the list. Based on the word's length, the list of words has been divided into three categories in addition to the Arabic letters and symbols. Then, each category is stored in a separate dictionary table. Table shows the categories of the words in the list. The following items describe the created dictionary tables:

1. *Arabic Characters (Letters, digits, and symbols) Dictionary (LDaSD) Table*: this dictionary table consists of Arabic characters (letters, digits, and symbols) that are used in the standard Arabic texts Table 2. This dictionary is used to encode the stand-alone letters and symbols where each one is given a specific index. This index is concatenated with ‘0’ as prefix coding pattern. Moreover, this dictionary table is used to encode the new word that doesn’t found in other tables, letter by letter. The used code for each letter and symbol consists of ‘0’ as a prefix followed by six binary digits, reflecting the character's index in this dictionary table. The resulting code for each character is seven bits in length, while the Unicode for each is sixteen bits. The character’s code size of the proposed algorithm is only around 44% of the original character’s size.

Table 2. Arabic language letters with pronunciation in arabic and english.

Letter	English Name	Arabic Name	Letter	English Name	Arabic Name
ا	alif	ألف	ض	Dād	ضاد
ب	Bā’	باء	ط	Tā’	طاء
ت	tā’	تاء	ظ	Zā’	ظاء
ث	thā’	ثاء	ع	ᶜayn	عين
ج	djīm	جيم	غ	ghayn	غين
ح	Hā’	حاء	ف	fā’	فاء

2. *Short Words Encoding Dictionary (SWED) Table*: this table consists of words with two and three letters in length (Table 3). In this SWED table, we assigned a sixteen-bit code for each word, starting with the prefix ‘10’ followed by fourteen binary bits reflecting the word index in this dictionary table. If these words encoded are encoded by Unicode, the code length will be 32 and 48 bits for words with two and three letters length, respectively. Comparing the proposed encoding scheme with Unicode one, we achieved a 50% and 33% size reduction for the words with two and three letters length, respectively. According to the used coding system for this dictionary table, the total number of entries that can be encoded using 14 bits code is 2^{14}=16384 *words*, which is less than the total number of words in this category which is 10617 (Table 1).
3. *Medium Words Encoding Dictionary (MWED) Table*: words with four to six letters length are grouped in this encoding dictionary table Table 4. The proposed algorithm encodes the words in this look-up table with a code length equal to 24 bits. The algorithm reserved ‘110’ as a prefix to encode the words and 21 bits to reflect the word index in the MWED table. The Unicode length for four, five- and six-word letters is 64, 80, and 96 bits. This means that the achieved code size reduction of the proposed algorithm compared to Unicode is 37%, 30%, and 25% for four, five, and six, respectively. In this dictionary table, the total number of entries that can be encoded is 2^{21}=2,097 *kilo−words*, while the number of Arabic words in this category is 1,587 Kilowords (Table 1).

Table 3. Sample of Short Words Encoding Dictionary (SWED) table.

Word	Pronunciation	Meaning
أب	'ab	Father
جد	jid	Grandfather
رث	ruth	Shabby
برد	bard	Cold
درس	daras	Lesson

Table 4. Sample of Medium Words Encoding Dictionary (MWED) table contents.

Word	Pronunciation	Meaning
آباء	aba’	Parents
بساط	bisat	Rug
حقيبة	haqiba	Bag
مسطرة	mustara	Ruler
تلفون	talifun	Telephone

Table 5. Sample of Long Words Encoding Dictionary (LWED) table contents.

Word	Pronunciation	Meaning
المهيمن	almuhaymin	Dominant
أأبصروه	’a’absaruh	See him
ساكنيها	sakiniha	Resident
جمهورية	jumhuria	Republic
ءأنذرتهم	’a’andhirathum	I warned them

4. *Long Words Encoding Dictionary (LWED) Table*: this table contains the remaining words with seven letters or more (Table 5). The used code for this table starts with ‘111’ followed by 23 binary digits. The code size for the words in LWED is 26 bits in length. The minimum size by the Unicode for words with seven letters is 112 bits which is greater than the proposed coded system by 4.3 times. The code size reduction for the words in this category is less than 23%.

The coding system used to encode the entries in LWED consists of 23 bits. Thus, the total number of entries that can be encoded is 2^{23}=8,388 *kilo−words*, while the number of words in this category is 7,598 kilowords (Table 1). Based on the proposed encoding scheme, which is shown in (Table 6), the total size of the encoding dictionary tables is large enough to have all Arabic words and have a free slot in each category to insert more new words. The following formula shows that the size of the encoding dictionaries consists of more than 10 million slots:

$$Dictionary\ size = 2^{23} + 2^{21} + 2^{14} + 2^{6} = 10{,}502{,}208\ slots \quad (1)$$

Table 6. Example 1. The coding sentence for an input text sentence with coding parts.

Prefix / Command Code	Symbol/word index code	Coding	Length	File number/ Command
10	00011001000000	بسم	14	2
110	00000001010001 1101110	الله	21	3
110	01001011011011 1011110	الرحمن	21	3
110	01001011011011 1100110	الرحيم	21	3
0100				Delete space

2.2. Arabic Text Message Encoding

After preparing the Arabic words dictionary tables, the input text (directly from the user or from an input text file) is encoded into a binary stream. The flowchart in Figure1 represents the developed algorithm to read the input text and convert it to a binary stream. The developed algorithm divides the input text into tokens. The token can be a word, a letter, or a symbol. This algorithm searches the dictionaries for the token; if it exists, the index of this token is returned. The encoding system converts the index into appropriate binary code and adds it to the binary stream.

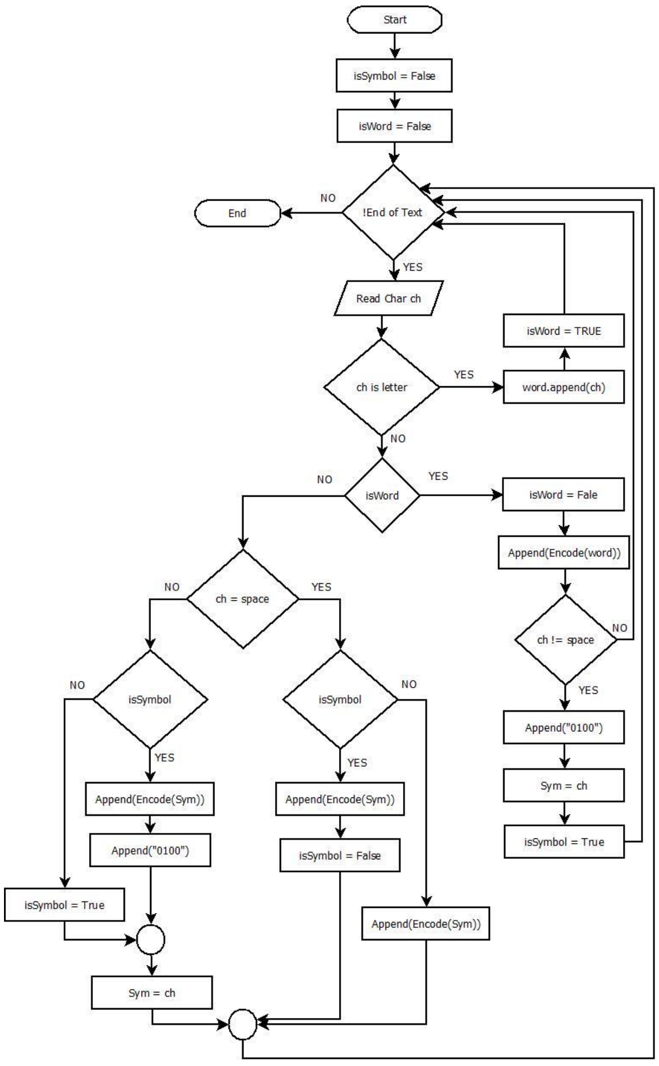


Figure 1. The algorithm for reading the text and building the corresponding binary stream.

In the case where the token does not find in the encoding dictionaries. Then, this token is manipulated according to one of the two possible cases:

1. First case: the word has a prefix, and/or postfix or both. The algorithm identifies the word, the prefix, and/or the postfix. Then, the algorithm encodes the prefix followed by the code of the word and encodes the postfix.
2. Second case: the word doesn't exist in the dictionary and cannot be divided into prefix, word, and/or postfix. Then, the code "0101" is inserted to identify that the code for this word is generated letter by letter. After finishing all letters, the code "0110" is inserted to indicate the end word code.

In formal writing, each word is followed by a space. Therefore, the space code wasn't added to the code text. The decoded system adds a space after each word directly. In the cases where the word is followed by any symbol other than space, a special code which is "0100", is added to inform the algorithm to cancel space addition. Figure 2 depicts the flowchart that encodes a text token (word) according to the proposed encoding algorithm.

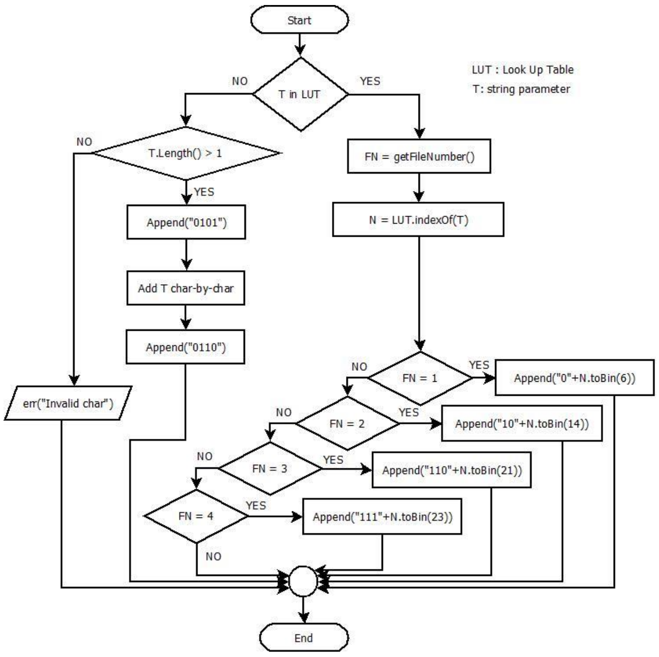


Figure 2. AddWord(T): This function implements the encoding algorithm which takes the string T and finds the proper code based on the indices in the dictionary.

Table 7. Binary representation and indexing for dictionary files.

Category Name	Prefix (binary)	Code length	Total number of slots in dictionary
Letters and Symbols	(0)2	6	26 Items
Short Words 2 and 3 letters length	(10)2	14	214 Items
Medium Words 4, 5 and 6 letters length	(110)2	21	221 Items
Long Words 7 letters length and more	(111)2	23	223 Items

Table 7 shows an encoding example for the Arabic input sentence ("بسم الله الرحمن الرحيم"). The complete binary code for the string is 100001100100000011000000000101000111011101100 100101101101110111101100100101101101111100110 0100. The first column consists of the category code or command code. The second column contains the binary representation of the word index in the dictionary, where the length of the code depends on the word category, which is shown in the last column. In Table 8, another example is represented. In this example, the last word is followed by a dot directly. Therefore, after the word code, the algorithm added the “not to add space” command code.

Table 8. The compression ratio for the testing files.

File	Original file size (bytes)	ATECA compressed File Size (bytes)	ATECA Compression Ratio
Quran	746,346	209,720	28%
Rand	1,542,380	291,702	19%
Alandalus1	46,536	14,012	30%
Alandalus2	4,656	1,152	25%

2.3. Mapping Binary Stream to Two-Dimensional Binary Matrix

In the two-dimensional binary matrix, shuffling operation becomes more effective and achieves its goal faster than shuffling a sequential stream. Of course, The main goal of shuffling is to separate adjacent bits away from each other and destroy any statistical feature that could be utilized to reveal the encrypted text. Thus, the sequential binary stream is converted to a two-dimensional binary matrix. The following steps are applied to convert the binary stream into the two-dimensional binary matrix where the size of the matrix can hold the binary stream with minimum padding:

1. Let nb equal to the size of binary stream in bits.
2. Calculate the size of the square matrix (N, N) that can hold the binary stream. Thus,

$$N = \lceil \sqrt{(ab)} \rceil \tag{2}$$

3. Define R and C as the two dimensions of the two-dimensional matrix. let R=C=N.
4. Let R=R − 1
5. Let C=C + 1
6. While R×C > nb , repeat 4
7. R=R+1, and C=C-1, then Define the two dimensional matrix TDB[R, C]
8. Fill the TDB matrix by the binary stream row by row.
9. Fill the unfilled cells of the last row with zeros.

Although shuffling operations might have better results utilizing a square matrix, the compression level achieved using a rectangular one is better. It will increase the complexity of the encryption algorithm through the need to guess the matrix size. Figure 3 represents the matrix filled by the code from example 1 (Table 6). In the last row, the code was padded by 0’s.

1	0	0	0	0	1	1	0	0	1	0	0
0	0	0	0	1	1	0	0	0	0	0	0
0	0	1	0	1	0	0	0	1	1	1	0
1	1	1	0	1	1	0	0	1	0	0	1
0	1	1	0	1	1	0	1	1	1	0	1
1	1	1	0	1	1	0	0	1	0	0	1
0	1	1	0	1	1	0	1	1	1	1	0
0	1	1	0	0	1	0	0	0	0	0	0

Figure 3. The matrix representation for example 1 (Table 6).

2.4. Two-Dimensional Shuffling Operations

As discussed earlier, the first step of the Arabic Text Encryption/compression algorithm is the encoding process, which generates a binary stream representing the original Arabic text. Then, the algorithm maps the binary stream to a two-dimensional binary matrix. Finally, the encryption/compression algorithm shuffles the two-dimensional matrix to separate adjacent bits away from each other. The separation distance and direction are random.

```
Algorithm 1: Horizontal Rotation function (HRF): 0 Function
Input: D: Decimal digit; M: The two-dimensional binary
        matrix TDB with size n∗m;
Output: M: Two-dimension coded matrix after ’0
          Function’ is implemented.
begin {main}
    dir=D mod 2; // The initial rotation direction; 0
    for rotate right and 1 for rotate left
    row=0; // initialize the row number by zero
    while row<n do // While not the end of the matrix
        call rotateRow(dir,row,D,M);
        row=row+1;
        if dir = 0 then // Switch the rotation direction
            dir = 1; // put the rotation direction to left
        else
            Dir = 0; // put the rotation direction to
            right
End {main}
```

Moreover, the sequence of running the shuffling functions is random via applying features of the proposed encryption key. This proposed strategy in developing the encryption algorithm makes discovering the encrypted text too hard. The randomness of separation of adjacent bits is achieved by utilizing a randomly generated encryption key. The encryption key is a decimal number with a variable length not less than twenty digits. The encryption key is generated randomly and updated regularly to ensure that it is not unrevealed.

The proposed encryption/compression algorithm utilizes four binary shufflings: horizontal rotation, vertical rotation, diagonal rotation, and circulation. Each function has inputs to determine the shuffling direction and shuffling distance. The proposed encryption/compression algorithm passes one of the encryption key digits to the shuffling functions as an input. The following subsection discusses these shuffling functions in detail.

2.4.1. Vertical Rotation Function (VRF):1 Function

Vertical Rotation Function (VRF), which has been assigned as “1 function,” rotates columns of the generated binary matrix. Algorithm (2) shows the pseudo-code for this function where the proposed algorithm passed one of the encryption digits to this function. VRF rotates columns up and down alternatively. The rotation direction of the first column is defined based on the input digit. Moreover, the input digits value represents the number of rotating bits.

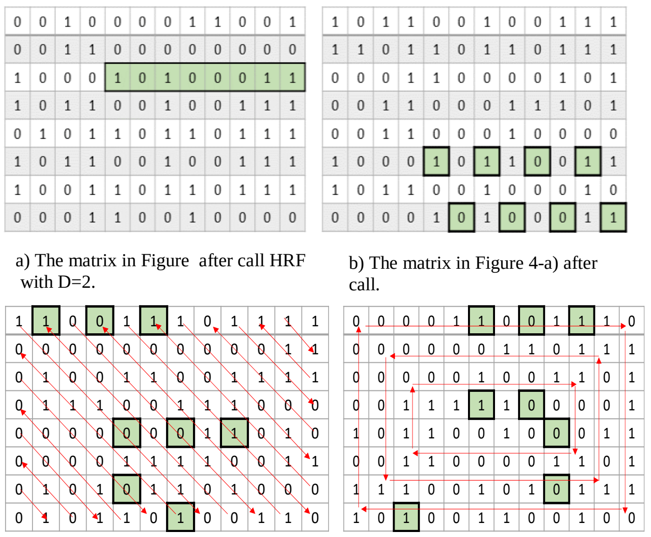
```
Algorithm 2: Vertical Rotation function (VRF):1 Function
Input: D: Decimal digit; M: The two-dimensional binary
          matrix TDB with size n∗m;
Output: M: Two-dimension coded matrix after '1
             Function' is implemented.
1    begin {main}
2        dir=D mod 2; // The initial rotation direction; 0
         for rotate up and 1 for rotate down
3        col = 0; // initialize the row number by zero
4        while col < n do // While not the end of the matrix
5            call rotateCol(dir, col, D, M);
6            col = col + 1;
7            if dir = 0 then // Switch the rotation direction
8                dir = 1; // put the rotation direction to
                 down
9            else
10               dir = 0; // put the rotation direction to up
11   End {main}
```

Figure 4-b illustrates the result of applying VRF on the matrix in Figure 4-a where the input digit Equals 3.

```
Algorithm 3: Diagonal rotation Function (DRF): 2 Function
Input: D: Decimal digit; M: The two-dimensional binary
          matrix TDB with size n∗m;
Output: M: Two-dimension coded matrix after '2 Function'
             is implemented.
1    dir = D mod 4 ; // The initial rotation direction
     // Category 1: 0 for rotate right-up and 3 for rotate left-
     down
     // Category 2: 1 for rotate right-down and 4 for rotate
     left-up
2    begin {main}
3        finish = false;
4        if dir = 0 or dir = 3 then // To set the index for the
         initial diagonal to start rotate with
5            i = 0; j = 1; //(0, j) is the index of top-left cell
             in the diagonal.
6            while !finish do // Repeat until all diagonal in
             matrix are rotated.
7                call rotateD(I, j, D, dir, M); // Rotate the
                 diagonal in dir direction, where cell (i,j) is
                 the top-right cell in the diagonal
8                if $j < m - 1$ then
9                    j = j + 1;
10               Else
11                   if $i \le n - 1$ then
12                       i = i + 1;
13               if dir = 0 then // Switch the direction
                 between right-up and left-down
14                   dir = 3;
15               else
16                   dir = 0;
17               if $i \ge n$ then
18                   finish = True;
19       else
20           if dir = 1 or dir = 2 then // We rotate
             diagonals in two directions right-down and
             left-up.
21               i = 0; j = m - 1; //(0, j) is the index of top-
                 right cell in the diagonal.
22               while !finish do // Repeat until all
                 diagonal in matrix are rotated
23                   callrotateD(I, j, D, dir, M); // Rotate
                     the diagonal in dir direction, where cell
                     (i,j) is the top-let cell in the diagonal.
24                   if j > 0 then
25                       j = j + 1;
26                   else
27                       if $i \le n - 1$ then
28                           i = i + 1;
29                   if dir = 0 then
30                       dir = 3;
31                   else
32                       dir = 0;
33                   if $i \ge n$ then
34                       finish = True;
35   end {main}
```

2.4.2. Diagonal Rotation Function (DRF): 2 Function

The Diagonal Rotation function (DRF) is similar to HRT and VRT, where the rotation is in the diagonal axis's direction. Therefore, we have four-direction options; right-up, right-down, left-up, and left-down. Thus, the rotation directions are ordered and assigned code as follows: right-up=0, right-down=1, left-up=2, and left-down=3. The input digit identifies the first direction. Then the direction is executed in order according to (right-up and left-down) or (right-down and left-up). Algorithm (3) depicts the pseudo-code for DRF, and Figure 4-c) represents an example of applying DRF where input digit D is equal to 5.


a) The matrix in Figure after call HRF with D=2.

b) The matrix in Figure 4-a) after call.

c) The matrix in Figure 4-b) after call DRF with D=5.

d) The matrix in Figure 4-c) after call CHF with D=4.

Figure 4. An example of implementing the four functions in order on the matrix in Figure 3.

2.4.3. Circulation Shuffling Function (CHF): 3 Function

The Circulation Shuffling Function (CHF) forms a ring from the outer rows and columns, shown in Figure 4-d. CHF circulates the bits in the ring either clockwise or counterclockwise n bits. The number of circulation bits n is an input for the function. Then it forms next circulation ring is formed from the next set of rows and columns. CHF applies the circulation where the direction is opposite to the direction of the first ring. Then, the operation is repeated until all columns and rows are included in one of the formed rings. Algorithm (4) shows the pseudo-code for this function. Figure 4-d) represents an example of applying CHF on the matrix in Figure 4-c) where input digit D Equals 4.

```
Algorithm 4: Circulation Shuffling Function (CHF): 3 Function
Input: D: Decimal digit; M: The two-dimensional binary matrix TDB with size n∗m;
Output: M: Two-dimension coded matrix after '3 Function' is implemented.
begin {main}
    dir = D mod 2; // The initial rotation direction: 0 for rotate clockwise, and 1 for rotate counter-clockwise
    number_of_rings = min($\lfloor \frac{n}{2} \rfloor$, $\lfloor \frac{m}{2} \rfloor$); // compute the number of rings that the matrix can have.
    index = 0;
    while index < number_of_rings do // While still has rings to manipulate
        Call circulate(dir, index, D, M) ;// circulate function circulates the ring from cell(index,index) in matrix M with direction (dir) by D times.
        index = index +1;
        if dir = 0 then // Switch the circulation direction
            dir =1 ; // put the circulation direction to counter-clockwise
        else
            dir = 0 ; // put the circulation direction to clockwise
end {main}
```

2.4.4. Proposed Encryption Key and Shuffling Operation

One of the main elements which strengthen the proposed encryption/compression algorithm is the encryption key. If the encryption is selected and utilized properly, the algorithm becomes more immune against most attacks and increases the time needed to break the algorithm. In our proposed algorithm, the encryption key has a variable length with a minimum digit. Also, the encryption key is generated randomly and needs to be updated regularly. These three proprieties weaken intruders tries to predict and or deduct it. The encryption key is transmitted between sender and receiver periodically in a secure channel. The key consists of a decimal number of digits (0 to 9). Let us define it as follows:

$$EK = D_0D_1D_2D_3D_4 \cdots D_n, wher\ n \geq 30 \tag{3}$$

The encryption key determines the behavior of encryption shuffling as follows:

```
Algorithm 5: Arabic Encryption/Compression Algorithm (ATECA).
Input: EK: Encryption Key; M: The two-dimensional binary matrix (TDB) with size n ∗ m;
Output: M: Two dimension binary matrix after shuffling operations are implemented.
begin {main}
    DD = $D_0D_1$; //The first two digits in the encryption key.
    rep = DD; //The number of repetitions should be at least 30 times.
    i = 0; //The index of the third digit in the EK to be used in shuffling.
    k = 2; //initialize k to 2 to pick the $D_2$ from encryption key.
    while i<rep do
        D = digit(k); //digit returns the digit $D_k$ in EK as in equation 1
        Func = D % 4; //The value of D determines the shuffling algorithm to be called.
        Switch func do
            case 0 do
                call HRF(D, M); break;
            case 1 do
                call VRF(D, M); break;
            case 2 do
                call DRF(D, M); break;
            case 3 do
                call CHF(D, M); break;
        end
        i = i + 1; k = (k+1) % length(EK); //Next repetition.
        if k = 0 then
            dir = 2;
    end
end {main}
```

1. The first two digits determine the number of repetitions for the main repetition loop, as shown in Algorithm (5). Thus, the repetition of the shuffling operations is variable, with a minimum number of repetitions is equal to 30. These features enhance the strength of the encryption algorithm because if the exact input text is encrypted differently in different periods, not only the encrypted output will be different, but also any statistical or semantic features will be completely different.
2. The encryption shuffling operation reads a digit from the encryption key (D= EK(m%EKs)), where m is the number of functions in each iteration, and EKs is the size of the encryption key. Then, the shuffling function is identified based on (D modulus 4), and the shuffling function utilizes D as a function input.

Finally, In the previous subsections, we have defined the main components of the proposed encryption algorithm: Arabic text encoding utilizing look-up dictionaries, Encryption key, and shuffle functions. Algorithm (5) illustrates the pseudo-code for the Arabic Text Encryption/Compression Algorithm (ATECA).

3. The Compression

In addition to Encryption, the proposed algorithm compresses the input text with a significant ratio. The proposed coding scheme causes the size of the encoded text to decrease sharply according to the percentages discussed in section 2.1. According to Arabic word size, the proposed algorithm achieved a compression level from 67% to less than 23% of the original word size. Figure 5 depicts compression level with respect t word size. In general, equation four is used to fin Compression Ratio (CR):

$$CR = \frac{Size\ of\ Compressed\ Data}{Data\ Size\ before\ Compressed} \tag{4}$$

This equation will measure the compression ratio for a set of Arabic text files and compared with some known compression algorithms. Table 9 shows the compressed rate for four tested files.

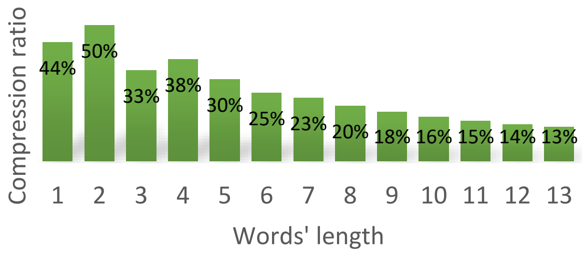


Figure 5. The percentages of the given code to each word based on word length to the Unicode size, which computed letter by letter.

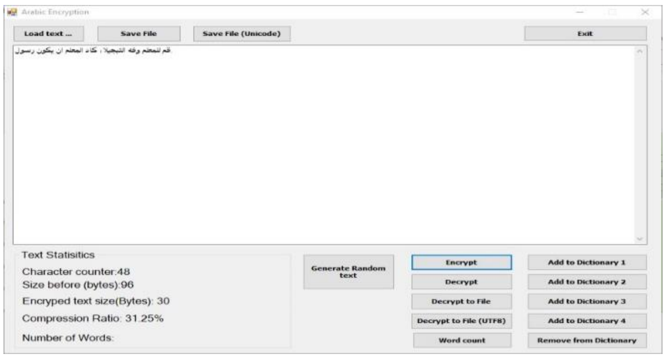
4. Implementation and Results Discussions

The proposed Encryption and compression algorithm has been implemented using C# language on Microsoft visual studio platform 2020 [23]. Figure 6 shows the interface program. The source code is available on: *https://github.com/ahmadaljarrah/ATECA*

For testing and evaluation of the proposed algorithm, we utilized the following four Arabic text files:

1. Quran.txt: this file contains the whole Holy Quran. The Holy Quran has the most formal Arabic words and is considered a reference for the Arabic language.
2. Andalus1.txt and Andalus2.txt: two files contain text from different resources about Andalus for different Authors. Andalus is the former Spain where the Arabic culture dominated that period.
3. Rand.txt: this file has been collected from different resources to have a large random Arabic text. The file content is intended to utilize currently Arabic text sentences and phrases.

These files are encrypted using the implementation of the proposed Arabic text encryption/compression algorithm, as its interface implementation is shown in Figure 6. The compression results for the four testing files are shown in Table 9. This table shows that the sizes of the four files are sharply decreased.



قم للمعلم وفه التبجيلا ، كاد المعلم ان يكون رسول.

Text Statisitics
Character counter:48
Size before (bytes):96
Encryped text size(Bytes): 30
Compression Ratio: 31.25%
Number of Words:

Figure 6. The main user interface for the proposed encryption/ compression algorithm implementation.

The proposed algorithm results are discussed in two fields, including compression and encryption. The compression is discussed by comparing the results with known compression techniques. The encryption is discussed by analyzing the strength of the proposed encryption algorithm.

4.1. Compression Analysis

The algorithm results have been compared to known algorithms in compression including, ZIP and RAR [22]. Table 9 shows the compressed rate for four tested files for ZIP, RAR, and our proposed ATECA (Figure 7). The results ensure the ATECA superiority when compared with ZIP in all tested files. Moreover, ATECA achieved a better compression ratio for the Andalus2 file if compared with the RAR technique.

Table 9. Comparisons of ATECA with known compression algorithms.

Test File	Rand	Quran	Alandalus 1	Alandalus 2
Original File Size (Bytes)	1,542,380	746,346	46,536	4,656
ZIP File Size	562,311	212,992	14,333	1,581
ZIP Comp. Ratio	36%	29%	31%	34%
RAR File Size	196,782	155,768	13,180	1,472
RAR Comp. Ratio	13%	21%	28%	32%
ATECA Compression File Size	291,702	209,720	14,012	1,152
ATECA Comp. Ratio	19%	28%	30%	25%

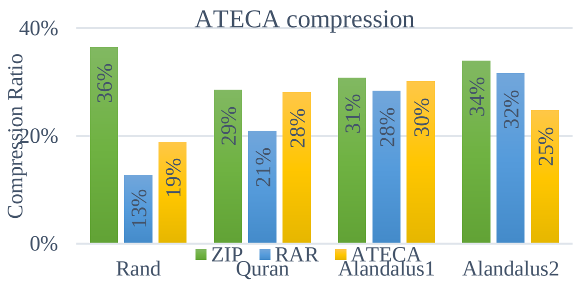


Figure 7. The compression ratio for the ACA, ZIP, and RAR algorithms.

The strength of the encryption algorithm is measured by the excellent design that makes discovering the encrypted text too hard to reveal from unauthorized users. The following points describe the strength of

ATECA design:

1. The input Arabic text is encoded utilizing Arabic word dictionary tables where the encoding unit is the Arabic word.
2. The index size for Arabic words is not fixed and is not a multiple of a byte.
3. The shuffling operations are applied on a two-dimensional binary matrix where the size of the matrix is a function of an encoded binary stream.
4. Each function receives a parameter where its' value is a decimal digit (0 to 9). the input is randomized via utilizing a random encryption key.
5. Each function shuffle or swap the matrix according to the received parameter. Each received value has a different effect.
6. The algorithm consists of four functions where each should be executed at least ninety-nine times.
7. The sequence of functions is random based on the randomness of the encryption key.
8. The shuffling outcome is cumulative. The output of the current shuffling function will be the input for the following shuffling function.
9. The encryption key length is variable, with a minimum number of digits equal to 30.

According to the mentioned facts, each function shuffles the matrix contents into twenty possible permutations. The input for each shuffling function is a decimal digit which means that we have ten possible values that lead to ten possible outputs. Moreover, the shuffling function has an input for the direction. Therefore, the number of possible permutations for each shuffling operation is 20. To calculate the number of possible permutations for the two-dimensional binary matrix, let us consider the following two factors:

1. The number of executed shuffling functions: As discussed earlier, each shuffling function has twenty possible permutations. Assume that the number of executed shuffling functions is two. Then, the number of permutations can be generated using the two shuffling functions where the second function depends on the first function output is $20*20 = 200 = 2*10^2$.
2. In the ATECA algorithm, the total number of functions to be executed depends on the repetition loop. The value of the repetition loop depends on the first two digits of the encryption key with a minimum value equal to 30. therefore, the repetition loop parameter will be between 30 and 99. Assuming that the value of repetition is 30, then the number of shuffling functions to be executed are 30*4 functions. Thus, the total number of permutations that can be generated is 20^{30*4}.

To generalize, the number of possible permutations for the proposed ATCEA P is as follow:

$$P = (PF)^{F\times K} \tag{5}$$

Where *PF* represents the number of permutations for each shuffling function, *F* is the number of functions, and *K* is the repetition shuffling loop. The repetition of the shuffling loop should be more than or equal to 30. We know that the number of shuffling functions is 4, and the number of permutations for each shuffling function is equal to 20. Let us assume the repetition loop k is equal to 30. Then, the total number of possible permutations for the two-dimensional binary matrix is:

$$\begin{aligned} P &= (20)_{4\times 30} \\ &= 1.3292279957849 \times 10^{156} \end{aligned} \tag{6}$$

The Probability of Breaking the Proposed ATECA: The probability of breaking the proposed ATECA algorithm relies on many factors such as:

- The encryption key: the attackers' ability to guess the utilized encryption key increases as their knowledge about encryption key features decreases. Because the length of the encryption key is variable, we can compute a lower bound for predicting the encryption key. The lower bound of the probability of guessing the encryption key is equal to the probability of getting the encryption key when the length is 20 (minimum encryption key length). the probability of getting the first digit in the encryption key is equal to 1/10. Then, the upper bound to predict the encryption key (Pkey) is:

$$PKey \leq 10^{20} \tag{7}$$

- The order of shuffling functions: the sequence of shuffling function is randomized so that sequence cannot be predicted from analyzing the encrypted text. This randomization vanishes any statistical features that would be discovered.
- Two-dimensional matrix size: the two-dimensional binary matrix is a function of the binary stream size. For example, adding a single statement at the end input text will produce different matrix dimensions. Changes in the matrix dimensions will change the behavior of the encryption algorithm entirely, and the output of the encryption for the new text is entirely uncorrelated to the output of the original text.

Assuming that the attacker knows the key length (assume 30 digits) and the exact process is done by each of the shuffling functions, the number of attempts to recover the text would be 10^{30} attempts.

Then let us assume that each loop in the encryption algorithm requires 1 ms; the total time to try all possibilities is:

$$\begin{aligned} time &\geq 10^{-3} \times 10^{20} = 10^{17} second \\ &\geq 3.215 \times 10^{9} years \end{aligned} \tag{8}$$

Thus, after assuming that the attacker has all knowledge to break the proposed ATECA algorithm except to predict the encryption key, it needs thousands of centuries to decrypt one message.

Furthermore, this algorithm would be extended to other languages and introduce different logical and arithmetical shuffling operations for two or more dimensions.

5. Conclusions

This paper presents a new technique to compress and encrypt Arabic text, which encodes the Arabic text utilizing Arabic word dictionaries, where each word is encoded by its index in the dictionary. Then, the encoded binary stream is mapped into a two-dimensional binary matrix. After that, the two-dimensional binary matrix is shuffled randomly utilizing four logical shuffling functions. The utilized shuffling functions are designed to maximize the separation distance between adjacent bits in a random fashion. The randomness of the shuffling is inherited from the randomly generated encryption key. The proposed algorithm utilizes a variable-length randomly generated encryption key. This algorithm has been implemented and tested on a wide range of input Arabic text documents. The results showed the ability of this algorithm to separate adjacent bits away from each other in a few operations and in a random fashion. Moreover, the proposed algorithm results showed its ability to compress input text by less than 30%. Finally, a mathematical analysis of the proposed algorithm showed that the proposed algorithm could be broken after $3.215*10^9$ years utilizing powerful computers.

Arabic script contains numerous diacritics (Tashkeel), therefore, in our future work we will extend the proposed algorithm to consider Arabic text with diacritics (Tashkeel). The other enhancement will take care of the required time to generate the encrypted text and to decrypt it back. The extended algorithm will use parallel to enhance the time complexity of the proposed algorithm.

References

[1] Abduljabbar R., Hamid O., and Alhyani N., "Features of Genetic Algorithm for Plain Text Encryption," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 1, pp. 434-441, 2021.

[2] Abduljabbar R., "Fast Approach for Arabic Text Encryption Using Genetic Algorithm," *European Journal of Scientific Research*, vol. 144, no. 4, pp. 342-348, 2017.

[3] Al-Bsharat A., "Developing a Word-Based Encryption Algorithm," Master Thesis, Yarmouk University, 2016.

[4] Al-Hazaimeh O., Al-Jamal M., Bawaneh M., Alhindawi N., and Hamdoni B., "A New Image Encryption Scheme Using Dual Chaotic Map Synchronization," *The International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 95-102, 2021.

[5] Al-Jarrah A., Al-bsharat A., and Al-Jarrah M., "Word-based Encryption Algorithm Using Dictionary Indexing with Variable Encryption Key Length," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 1, pp. 669-683, 2022.

[6] Al-Omari A., "ABJAD Arabic-Based Encryption," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, pp. 164-170, 2018.

[7] Alla K. and Ramachandran V., "A Novel Encryption Using Genetic Algorithms and Quantum Computing with Roulette Wheel Algorithm for Secret Key Generation," *in Proceeding of the Analysis and Applications*, Singapore, pp. 263-271, 2020.

[8] Alqahtani Y., Kuppuswamy P., and Shah S., "New Approach of Arabic Encryption/Decryption Technique Using Vigenere Chiper on Mod 39," *International Journal of Advanced Research in IT and Engineering*, vol. 2, no. 12, pp. 1-9, 2013.

[9] Alsuhibany S., "Developing a Visual Cryptography Tool for Arabic Text," *IEEE Access*, vol. 7, pp. 76573-76579, 2019.

[10] Attia M., "Arabic Wordlist for Spellchecking." https://sourceforge.net/projects/arabic-wordlist/, Last Visited, 2019.

[11] Aysan M. and Kuppuswamy p., "Hybrid Combination of Message Encryption Techniques on Arabic Text Using New Symmetric Key and Simple Logarithm Function," *International Journal of Scientific Knowledge Computing and Information Technology*, vol. 5, pp. 37-41, 2014.

[12] Fatoum A., "Wordlists for Arabic," https://github.com/a3f/arabic-wordlists, Last Vistited, 2022.

[13] Freeman W. and Miller E., "An Experimental Analysis of Cryptographic Overhead in Performance-Critical Systems," *in Proceeding of the 7th International Symposium on Modeling Analysis and Simulation of Computer and Telecommunication Systems*, College Park, pp. 348-357, 1999.

[14] Green W., "Integrity in Information Systems," *Integrity and Internal Control in Information Systems: Volume 1: Increasing the confidence in Information Systems*, pp. 295, 2013.

[15] Habeeb R., "Arabic Text Cryptanalysis Using Genetic Algorithm," *Iraqi Journal for Electrical and Electronic Engineering*, vol. 12, no. 2, pp. 161-166, 2016.

[16] Hwang S., "Security Flaws of Off-Line Micro Payment Scheme with Dual Signatures," *in Proceeding of the Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, Netherlands, pp. 905-909, 2014.

[17] Karima A., Zakaria E., Yamina T., Mohammed S., Selvam P., and Venkatakrishnan V., "Arabic Text Categorization: a Comparative Study of Different

Representation Modes," *Journal of Theoretical and Applied Information Technology*, vol. 38, no. 1, pp. 1-5, 2012.

[18] Kataria S., Kumar T., Singh K., and Nehra S., "ECR (Encryption with Cover Text and Reordering) Based Text Steganography," *in Proceeding of the IEEE 2nd International Conference on Image Information Processing*, pp. 612-616, 2013.

[19] Khare R. and Raghuwanshi K., "A Review of Video Steganography Methods," *International Journal of Research in Advent Technology*, vol. 2, no. 1, 2014.

[20] Kuppuswamy P. and Alqahtani Y., "New Innovation of Arabic Language Encryption Technique Using New Symmetric Key Algorithm," *International Journal of Advances in Engineering and Technology*, vol. 7, no. 1, pp. 30, 2014.

[21] Mahmoud A. and Zrigui M., "Semantic Similarity Analysis for Corpus Development and Paraphrase Detection in Arabic," *The International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 1-7, 2021.

[22] Marton Y., Wu N., and Hellerstein L., "On Compression-based Text Classification," *in Proceeding of the European Conference on Information Retrieval*, Santiago de Compostela, pp. 300-314, 2005.

[23] Microsoft, "Microsoft Visual Studio," https://visualstudio.microsoft.com/, Last Visited, 2020.

[24] Pal D., Raghavendra P., and Babu A., "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.

[25] Rihan S. and Osma S., "Arabic Cryptography Technique Using Neural Network and Genetic Algorithm," *International Research Journal of Computer Science*, vol. 3, pp. 35-42, 2016.

[26] Roy S., "The Extensive Bit-level Encryption System (EBES)," *International Journal of Information Technology and Computer Science*, vol. 5, no. 5, pp. 67-73, 2013.

[27] Shaban S., "A New Algorithm for Encrypting Arabic Text Using the Mathematical Equation," *Diyala Journal of Engineering sciences*, vol. 10, no. 1, pp. 21-30, 2017.

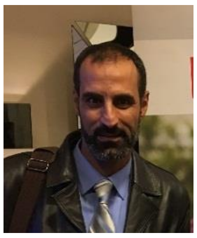
[28] Shareef R., Al-Shakarchy D., Abd H., Al-Nasrawi A., Al-Shahad F., and Aleqabie J., "New Cryptographic System of Romanized Arabic Text Based on Modified Playfiar," *Journal of Engineering and Applied Sciences*, vol. 14, no. 4, pp. 1331-1338, 2019.

[29] Suthanthiramani P., Muthurajkumar S., Sannasi G., and Arputharaj K., "Secured Data Storage and Retrieval Using Elliptic Curve Cryptography in Cloud," *The International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 56-66, 2021.

[30] Thahab A., "A Novel Secure Video Steganography Technique Using Temporal Lifted Wavelet Transform and Human Vision Properties," *The International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 147-153, 2020.

[31] Wang S. and Liu G., "File Encryption and Decryption System Based on RSA Algorithm," *in Proceeding of the International Conference on*, Chengdu, pp. 797-800, 2011.

[32] Waters B., "Functional Encryption: Origins and Recent Developments," *in Proceeding of the International Workshop on Public Key Cryptography*, Japan, pp. 51-54, 2013.



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.

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.

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.