Computer Science and Information Technologies
Vol. 3, No. 1, March 2022, pp. 51~64
ISSN: 2722-3221, DOI: 10.11591/csit.v3i1.p51-64

Modified Playfair cryptosystem for improved data security

Esau Taiwo Oladipupo[1], Oluwakemi Christiana Abikoye[2]
[1]Department of Computer Science, Federal Polytechnic Bida, Bida, Nigeria
[2]Department of Computer Science, University of Ilorin, Ilorin, Nigeria

Article Info

Article history:

Received Dec 3, 2021
Revised Jan 22, 2022
Accepted Feb 09, 2022

Keywords:

Brute force attack
Confusion
Cryptanalysis
Diffusion
Frequency analysis

ABSTRACT

Playfair is the earliest known classical block cipher which is capable of taking two characters as a unit in the process of encryption and decryption. However, the cipher is suffering from vulnerability to many cryptanalysis attacks due to a lack of confusion and diffusion properties, an inability to handle numbers and special characters in the process of encryption and decryption, and a host of other deficiencies. Although several modifications and improvements had been done by different researchers, the emphasis has been on the modification of the key matrix to accommodate more characters to increase the keyspace. No attention has been given to increment in the size of the block that the Playfair cipher can handle at a time. In this paper, a modified Playfair (MPF) cryptosystem that is capable of handling different block sizes with high diffusion and confusion properties is developed. cryptanalysis of the developed cryptosystem was carried out and the results show that the MPF cryptosystem is resistant to Known plaintext attack, chosen-plaintext attack, chosen ciphertext attack, frequency analysis attack, autocorrelation attack, differential cryptanalysis attacks, entropy attacks, brute force attack, and can handle variable block sizes.

Corresponding Author:

Esau Taiwo Oladipupo
Department of Computer Science, Federal Polytechnic Bida
KM 1.5 Doko Road, 912211, Bida, Nigeria
Email: taiwotheophilus@gmail.com

1. INTRODUCTION

Diffusion and confusion properties are two desired attributes of a cipher that make it difficult to be broken by the use of statistical analysis [1], these properties are lacking in classical ciphers [2], [3]. The keyspace of the classical ciphers is also limited and this limitation makes it easy for the present-day computers which are equipped with parallelism, pipelining, and multiprocessing properties to break these ciphers in seconds using brute force attack. All these weaknesses summed up, made the researchers consider classical ciphers as weak and irrelevant for securing information in this present age [4]. However, most of the celebrated modern cryptography algorithms today are just improvements over these classical ciphers. For instance, one of the earliest known block ciphers is Playfair [5] but today we have several block ciphers. This reality has made many researchers such as [2], [6]-[11] and a host of other researchers worked on some of these classical ciphers in order to improve their security and thereby make them relevant in this information age.

Many research works have been done on the improvement of the Playfair cipher. [12] and [13] implemented a 5x5 and an 8x8 square matrix key Playfair cipher respectively using a linear feedback shift register (LFSR). In the case of 8x8 square matrix key was used on DNA-encoded data. [9] modified 5x5 square matrix key Playfair cipher into a 7x4. [4] refined 5x5 square matrix key Playfair cipher into a 16x16 square matrix key Playfair cipher.

From the above discussion, it is obvious that a lot of work has been done in the modification of the Playfair cipher. However, attention has been on the modification of the size of the square key used in the Playfair cipher. None of the work in the literature review has made any attempt to increase the block size of the Playfair cipher from 2 characters it has been from inception. In this paper, the effort has been made to modify the existing 16 x 16 square matrix key Playfair cipher to handle variable block sizes. In addition, bit modification is introduced into the design of the developed modified Playfair cipher in order to introduce diffusion and confusion properties which are lacking in the existing Playfair cipher.

The rest of the paper is organized as follows. In section 2, an overview of the Playfair cipher and a review of related works are discussed. Section 3 explains the design of the modified Playfair cryptosystem (MPF). The results and discussion for different experiments conducted on MPF are given in section 4. Section 5 gives the conclusion of the research work.

2. OVERVIEW OF PLAYFAIR CIPHER AND REVIEW OF RELATED WORK

In this section overview of the Playfair cryptosystem and a review of some modification works on the cipher are discussed.

2.1. Overview of Playfair cipher

The Playfair cipher is an example of a polyalphabetic cipher and it was Charles Wheatstone who invented the cipher in 1854 but was named after lord Playfair who promoted its usage [14]. A polyalphabetic cipher treats a combination of two letters (digraphs) in the plaintext as a single unit and converts these digraphs into ciphertext digraphs using a key square. The key square is formed by writing a keyword horizontally with duplicate letters being removed. The rest of the square is filled with the remaining letters of the alphabet, in alphabetical order. For instance, if the keyword KEY is chosen as the key, the key square becomes

K	*E*	*Y*	*A*	*B*
C	*D*	*F*	*G*	*H*
I/J	*L*	*M*	*N*	*O*
P	*Q*	*R*	*S*	*T*
U	*V*	*W*	*X*	*Z*

Going by age of the cipher, mode of operation number of keys involved, and the number of characters that are being treated as a unit, the Playfair cipher can be classified to be an example of classical, substitution, symmetric, block cipher. Some peculiarity of the Playfair cipher as given by [15] include:

i. No plaintext letter can be represented in the cipher by itself.
ii. Any given letter can be represented by 5 other letters.
iii. Any given letter can represent 5 other letters.
iv. Any given letter cannot represent a letter that it combines with diagonally.
v. It is twice as probable that the two letters of any pair are at the corners of a rectangle than as in the same row or column.

A critical analysis of the Playfair cipher reveals the following weakness in the cipher:

i. The Playfair with 5 X 5 matrix takes I and J as one character, again if a message in pairs of letters ends as an odd number instead of even, X is added at the end but neglected for decryption [16]. These assumptions are not the best for a good cryptography algorithm.
ii. Only 26 letters can be used as the keyword. The keyspace is limited and this limitation makes the cipher an easy prey under brute force attacks in today's computer.
iii. Space between two words in the plaintext is not considered as one character;
iv. The cipher cannot use special characters and numbers;
v. Only uppercase alphabets are used for the encryption and decryption process
vi. Double letters in plaintext as a pair end up with an X separator whereas the letter X itself gets used as another recognized letter
vii. Due to pairwise encryption, the Playfair cipher is significantly harder to break than the Caesar cipher. Although the cryptanalysis of Playfair is considerably more difficult than that of monoalphabetic cipher, it is still possible with 600 possible digraphs. With 600 (25 x 24) possible digraphs, a considerably larger ciphertext is required in order to make it more difficult to break [1].
viii. Playfair cipher is vulnerable to statistical analysis attack and several successful attacks such as [17]-[19] have been reported.

2.2. Related work

Several attempts have been made by researchers to overcome the weaknesses of conventional Playfair ciphers. A 6 × 6 matrix was proposed by [20] instead of 5 × 5 the conventional Playfair cipher. The construction of the matrix key is similar to that of the conventional technique but with a larger set of alphabets. This matrix is large enough to accommodate numerical digits (0 to 9) in addition to 26 English alphabets in the classic technique. In addition, the I/J was counted as two separate letters and each is placed in separate cells in order to avoid ambiguity at decryption time. Similarly, [9] proposed a 7 x 4 matrix key Playfair cipher where two symbols '*' and '#" were added to create a one-to-one correspondence between plaintext and ciphertext A DNA and amino acids-based Playfair cipher algorithm where the user is capable of using any combination of alphabets, numbers, special characters, or even spaces in a plain-text was proposed by [21]. During the encryption process, the data is first represented in binary form, which is later transformed into sequences of DNA nucleotides. Subsequently, these nucleotides pass through a Playfair encryption process based on amino-acids structure.

Hamad *et al*. [14] proposed a 16 x 16 modified Playfair cipher for the encryption and decryption of images. An integer number, a key, which acts as the seed value in a random permutation module was used to randomly construct the substitution matrix after which the Playfair encryption process is applied to pair of pixels. The resultant scrambled image is then XORed with a randomly generated mask that has the same dimensions as the scrambled image. The reverse was carried out during the decryption process. The approach perfectly produced distinct cipher images even with similar keys and the decryption process obtained the original image from the cipher image. Hassoun *et al.* [22] used a bio-molecular technique to enhance the Playfair algorithm. A new key creation method out of a key using two secure lock-up tables for constructing the matrices for the encryption/decryption processes was suggested. The built system can be used in encrypting and decrypting big data.

An adaptive Playfair cryptography algorithm was proposed by [10]. The scheme employs the use of three keys and the general rule of the original Playfair cipher is followed. The modified encryption phase is depending on applying the odd pairs of the message to the first key matrix and applying the even pairs of the message to the second key matrix. Then the result is XORed with the third key. The decryption process reverses the encryption process to obtain the plaintext from the ciphertext.

In the reviewed literature the emphasis of the researchers has been on the modification of the Playfair matrix key. Either increase the number of characters or introduce the technique of permutations to ensure improvement over the conventional Playfair cipher. None of the existing works has ever discussed how the size of the block of Playfair cipher can be increased from 2 to other sizes. Therefore, in this paper, a modification to the existing Playfair cipher which will enable variable block sizes to be handled by the cipher is introduced.

3. DESIGN OF MODIFIED PLAYFAIR CRYPTOSYSTEM

3.1. Key generation process in modified Playfair cryptosystem

Key generation involves the generation of secret keys and Playfair keys. To generate a secret key in the MPF cryptosystem, only the block size is to be specified. The algorithm uses the supplied block size to generate a random number from which different information is derived to carry out the encryption and decryption process. The algorithm for generating the random key in modified Playfair MPF is given in Algorithm 1.

Algorithm 1: Pseudocode for MPF key generation process

```
Module MPFkeygen
Input
Block size: an integer
Output: key: an integer
Variables
Bitsize: an integer
1.  BEGIN
2.  GET blocksize
3.  CALCULATE Bitsize = blocksize * 8
4.  IF Bitsize < 128 THEN
     a.  DISPLAY error message
         ELSE
     b.  LET key = GENERATE a random number whose size is Bitsize
     c.  OUTPUT key
    ENDIF
5.  END
```

Playfair key is generated from the secret key generated in Algorithm 1. Playfair key is a 16 x 16 matrix of ASCII characters. Information is generated from the secret key obtained from Algorithm 1 to generate an array of random integers (the procedure for generating the random number is given in Algorithm 3) which forms the pattern to be used in the reshuffling of the ASCII characters before they are put in 16 x 16 matrix. Algorithm 2 describes the procedure for generating the Playfair key.

Algorithm 2: Pseudocode for generating Playfair key in modified Playfair cryptosystem

```
Module generatePlayfairkey (key)
Output: Playfairkey: ASCII character in 2D (16 x 16) array of characters
1.  BEGIN
2.  LET alphabet = [CHARACTER[i], FOR I = 0.. 255]
3.  LET keyspace = LENGTH(alphabet)
4.  LET confusionkey = CALL randgen(key, keyspace)
5.  LET playarray = CALL reshuffleforencryption(confusionkey,alphabet)
6.  LET Playfairkey = CONVERT playarray to 2D [1..16, 1..16]
7.  END
```

Algorithm 3: Pseudocode for generating array of random numbers in MPF cryptosystem

```
Module randomnumgenerator (seedn, keyspace)
Output: confusionkey: 1D array of random integers
1.  BEGIN
2.  INITIALISE random number generator using seedn
3.  LET confusionkey = GENERATE distinct keysplace random numbers
4.  END
```

3.2. Reshuffling of characters and bits of key, plaintext, and ciphertext characters/bits

As one of the measures to introduce confusion and diffusion in MPF Playfair key, plaintext and ciphertext are reshuffled. The permutation is done during encryption and decryption processes at both character and bit levels. Algorithm 4 and 5 describe how reshuffling of characters/bits are carried out in MPF during encryption and decryption processes respectively. An array of characters together with the pattern to be used for reshuffling is given to the function which produces the reshuffled array of characters/bits.

Algorithm 4: Pseudocode for reshuffling characters or bits in MPF

```
Module Reshuffleforencryption (Textmatrix, key)
Output Arrangedtext[1..C]: 1D array of characters
Variables
Integer: k, col, x, C
message[1..C]: arrays of characters
1.  BEGIN
2.  LET C = LENGTH(Textmatrix)
3.  INITIALIZE message as empty matrix
4.  LET k = 1
5.  FOR col = 1 TO C
6.  LET x = key[col]
7.  LET message( k) = TextMatrix[x]
8.  LET k = k + 1
9.  END FOR
10. LET arrangedtext = message
11. OUTPUT arrangedtext
12. END
```

Algorithm 5: Reshuffling of characters during the decryption process

```
Module Reshufflefordecryption (ciphertext, key)
Output: plaintext[1..C]: 1D array of characters
Variables
Integer: col, x, C
message[1..C]: arrays of characters
1.  BEGIN
2.  LET C = LENGTH(ciphertext)
3.  INITIALIZE message as empty matrix
4.  FOR col = 1 TO C
5.  LET x = key[col]
6.  LET message( x) = ciphertext[col]
7.  LET k = k + 1
8.  END
9.  LET plaintext = message
10. OUTPUT plaintext
11. END
```

3.3. Bit manipulation in MPF

The aim of introducing bit manipulation is to ensure proper confusion and diffusion in MPF. The bit modification process employs the use of bit grouping, bit permutation, and XOR operation to ensure the proper mixing of key and plaintext bits. Algorithm 6 describes bit modification during the encryption process while Algorithm 7 is the description of the reverse process of Algorithm 6 during the decryption process.

Algorithm 6: Pseudocode to perform bit manipulation during the encryption process

```
Module manipulatedbits = Bitmanipulation(blockbits, randseed)
Input: block of binary digits, randomseed
Output: manipulatedbits
1.  BEGIN
2.  t = CONVERT randseed to its binary form
3.  lt = LENGTH(t)
4. IF lt< 12 lt = 12 ELSE IF lt > 16 lt = 16
5.  Randkeys = GENERATE array of lt random seeds using randseed as seed
6.  Lnibble = empty string
7.  Rnibble = empty string
8.  bitsize = LENGTH (blockbits)
9.  FOR i = 1 to bitsize, taking 8 bits at a time
    i.    Abyte = blockbits(i:i+7)
    ii.   Lnibble = CONCATENATE (Lnibble, Abyte(1:4))
    iii.  Rnibble = CONCATENATE(Rnibble, Abyte(5:8))
    END FOR
10. FOR j = 1 to lt
    i.  Seed1 = Randkeys(j)
    ii. LET xornibble = Lnibble XOR Rnibble
    iii. LET Fullbyte = CONCATENATE(xornibble, Rnibble)
    iv. confusionkey = GENERATE bitsize random integer using seed1 as seed
    v.  transposedbits = CALL Reshuffleforencryption(Fullbyte, confusionkey)
    vi. LET Lnibble = transposedbits(1: bitsize/2)
    vii. LET Rnibble = transposedbits(bitsize/2 +1 : bitsize)
    END FOR
11. Manipulatedbits = transposedbits
12. END
```

Algorithm 7: Algorithm for reverse bit modification during the decryption process

```
Module blockbits = Reversebitmodification(binX, randseed)
Input: block of binary digits, randseed
Output: blockbits
1.  BEGIN
2.  t = CONVERT randseed to its binary form
3.  lt = LENGTH(t)
4.  IF lt< 12 lt = 12 ELSE IF lt > 16 lt = 16
5.  Randkeys = GENERATE array of lt random seeds using randseed as seed
6.  Bitsize = LENGTH(binX)
7.  reversedbinX = binX
8.  FOR i = lt to 1
    i.  seed1 = Randkeys(i)
    ii. confusionkey = GENERATE bitsize random integer using seed1 as seed
    iii. transposedbit = CALL Reshuffleforencryption(reversebinX, seed1) //algorithm 5c
    iv. LET lnibble = transposedbit(1: bitsize/2)
    v.  LET rnibble = transposedbit(bitsize/2 +1: bitsize)
    vi. LET xornibble = lnibble XOR rnibble
    vii. LET reversebinX = CONCATENATE(xornibble, rnibble)
    END FOR
9.  LET x = bitsize/2
10. LET m1 = reversebinX(1:x)
11. LET m2 = reversebinX(x+1:bitsize)
12. INITIALIZE msg as empty string
13. FOR j = 1 to x taking 4 bits at a time
    i.  LET L = m1(j:j+3)
    ii. LET R = m2(j:j+3)
    iii. LET msg = CONCATENATE(msg, L, R)
    END FOR
14. blockbits = msg
15. END
```

3.4. Application of conventional Playfair technique

MPF uses the conventional Playfair technique in the formation of the characters of the ciphertext from plaintext as well as the derivation of plaintext characters from the ciphertext. A block of characters together with the Playfair key is supplied to the module which carries out the conventional Playfair technique of encryption and decryption. Algorithms 8 and 9 describe the conventional Playfair technique during encryption and decryption processes respectively.

Algorithm 8: Pseudocode for encryption in Playfair cryptosystem

```
Module Playfairencryption(Blocktext, Playfairkey)
Output: ciphertext: string of characters of length C
1.  BEGIN
2.  Break the Blocktext into pairs of letters, refinedpair
3.  DO FOR each pair P in refinedpair to form ciphertext
    a.  If the two letters of P appear on the same row in the ksquare, replace each
        letter by the letter immediately to the right of it in the square (cycling
        round to the left hand side if necessary)
    b.  If the two letters appear in the same column in ksquare, replace each letter
        by the letter immediately below it in ksquare (cycling round to the top of the
        square if necessary
    c.  If the two letters of p are on different rows and columns, form rectangle for
        which the two letters of P are two opposite corners. Replace each letters in P
        with the letter that forms the other cornerner of the rectangle that lies on
        the same row as that of letter in P.
    END FOR
4.  OUTPUT ciphertext
5.  END
```

Algorithm 9: Pseudocode for decryption in Playfair cryptosystem

```
Module PlayfairDecryption
Input ciphertext: string of characters of length C
key: string of character with no repeating letters
Output: ciphertext: string of characters of length C
1.  BEGIN
2.  Generate key square ksquare
3.  Break the ciphertext into pairs of letters, refinedpair
4.  DO FOR each pair P in refinedpair to form plaintext
    a.  If the two letters of P appear on the same row in the ksquare, replace each
        letter by the letter immediately to the left of it in the square (cycling
        round to the right hand side if necessary)
    b.  If the two letters appear in the same column in ksquare, replace each letter
        by the letter immediately above it in ksquare (cycling round to the down of
        the square if necessary
    c.  If the two letters of p are on different rows and columns, form rectangle for
        which the two letters of P are two opposite corners. Replace each letters in P
        with the letter that forms the other corner of the rectangle that lies on the
        same row as that of letter in P.
    END FOR
5.  OUTPUT plaintext
6.  END
```

3.5. Key modification in MPF

No two blocks use the same key in MPF. The key that is used for the first block is modified and the newly generated key is used for the second. This process continues until all the blocks are treated. The key modification process employs the use of two modules. The first module in Algorithm 10 uses the information derived from the secret key to generate a seed that is used for the initialization of the random number generator. Algorithm 11 is the description of how a new key is generated from the previous one.

Algorithm 10: Generate random seed

```
Module genrandomseed (binarykey)
Output: randomseed
1.  BEGIN
2.  Blocksize = DETERMINE block size from the binarykey
3.  DETERMINE number of 1's and 0's in the binarykey, x1, x2
4.  x2 = x1 + Blocksize
5.  k = binarykey[x1…x2]
6.  k = CONVERT k to decimal number
7.  IF k is greater than 2^32 THEN k = k MOD 2^32 -1
8.  randomseed = k
9.  END
```

Algorithm 11: Generate new key

```
Module GETNEXTKEY (randseed, binarykey)
Output: newbinkey
1.  BEGIN
2.  x = LENGTH(binarykey)
3.  binseed = CONVERT randseed to binary number
4.  k = CONCATENATE(binseed, binarykey)
5.  newbinkey = k[1..x]
6.  END
```

3.6. Encryption and decryption modules in MPF

Each block of plaintext passes through the encryption module MPlayfair in order to generate ciphertext for the block. Algorithm 12 describes how a block of plaintext is encrypted in MPF.

Algorithm 12: Pseudocode for encryption module in modified Playfair cryptosystem

```
Module MPlayfair(blocktext, binkey, radno)
Output: encryptedblock: 1D array of ASCII characters
1.  BEGIN
2.  LET Playfairkey = CALL generatePlayfairkey(radno)
3.  LET binp = CONVERT blocktext to binary form
4.  LET binplayxor = binp XOR binkey
5.  LET mbinplayxor = CALL Bitmanipulation (binplayxor,radno)
6.  LET charplayxor = CONVERT mplayxor to ASCII characters
7.  LET encryptedblock = CALL Playfairencryption (charplayxor, Playfairkey)
8.  END
```

Each block of ciphertext passes through the decryption module DMPlayfair in order to retrieve the plaintext block from the ciphertext block. Algorithm 13 describes how a block of ciphertext is decrypted in MPF.

Algorithm 13: Pseudocode for decryption module in modified Playfair cryptosystem

```
Module DMPlayfair( blocktext, binkey, radno)
Output: decryptedblock: 1D array of ASCII characters
1.  BEGIN
2.  LET Playfairkey = CALL generatePlayfairkey(radno)
3.  Playdecryptedblock = CALL Playfairdecryption(Blocktext,Playfairkey)
4.  LET binp = CONVERT Playdecryptedblock to binary form
5.  LET mbinplayxor = CALL reversbitmanipulation (binp,radno)
6.  LET binplayxor = mbinplayxor XOR binkey
7.  LET charplayxor = CONVERT binplayxor to ASCII characters
8.  END
```

3.7. MPF encryption and decryption process

The MPF encryption process is depicted as shown in Figure 1. MPF takes plaintext and a secret key from the user as input. The block size is determined by the key. The plaintext is then divided into blocks B1, B2, …, Bn. The number of blocks n is determined and then the key is used to generate array S [S1, S2, …, Sn] of random integers of n elements. A new key is generated using module Getnextkey that takes the key and element Si from S. The newly generated key Ki, the plaintext block Bi and random number S[i] are then passed as parameter to module MPlayfair which produces the ciphertext Ci for the plaintext block Bi. The process repeats itself n number of times. The cipher blocks Ci where i = 1 to n is concatenated together to form the ciphertext. The description of the MPF encryption process is given in Algorithm 14.

The MPF decryption process is depicted as shown in Figure 2. The MPF decryption process takes ciphertext and secret key from the user as input. The block size is determined by the key. The ciphertext is then divided into blocks C1, C2, …, Cn. The number of blocks n is determined and then the key is used to generate array S[S1, S2, …, Sn] of random integers with n elements. A new key is generated using module Getnextkey that takes key and element Si from S. The newly generated key Ki, the ciphertext block Ci and random number S[i] are then passed as a parameter to module DMPlayfair which produces the plaintext Bi for the cipherblock block Ci. The process repeats itself n number of times. The plaintext blocks Bi where i = 1 to n are concatenated together to form the plaintext. Algorithm 15 describes the MPF decryption process.

Algorithm 14: Modified Playfair encryption process

```
Input: plaintext, key
Output: ciphertext
1.  BEGIN
2.  CONVERT key to its binary form binkey
3.  Blocksize = LENGTH(binkey)/8
4.  DETERMINE the number of block nblocks in the plaintext
5.  CONVERT key to its binary form binkey
6.  randomseed = CALL genrandomseed(binkey)
7.  INITIALIZE random number generator using randomseed
8.  Randomkeys[1..nblocks] = GENERATE array of size nblocks of random integers
9.  INITIALIZE encryptedtext as EMPTY STRING
10. FOR i = 1 TO LENGTH(plaintext) taking blocksize characters at a time DO
11. LET block = plaintext ( i to i + blocksize -1)
12. binkey = CALL GETNEXTKEY(binkey, Randomkeys(i))
13. mPlayfairblock = CALL MPlayfair( block, binkey, Randomkeys(i))
14. LET encryptedtext = CONCATENATE( encryptedtext, mPlayfairblock)
    END FOR
15. LET ciphertext = encryptedtext
16. OUTPUT ciphertext
17. END
```

Algorithm 15: Modified Playfair decryption process

```
Input: ciphertext, key
Output: plaintext
1.  BEGIN
2.  CONVERT key to its binary form binkey
3.  Blocksize = LENGTH(binkey)/8
4.  DETERMINE the number of block nblocks in the ciphertext
5.  randomseed = CALL genrandomseed(binkey)
6.  INITIALIZE random number generator using randomseed
7.  Randomkeys = GENERATE array of size nblocks of random integers
8.  LET Randomkeys = Randomkeys[1..nblocks]
9.  INITIALIZE encryptedtext as EMPTY STRING
10. FOR i = 1 TO LENGTH(ciphertext) taking blocksize characters at a time DO
    i.  LET block = ciphertext ( i to i + blocksize -1)
    ii. binkey = CALL GETNEXTKEY(binkey, Randomkeys(i))
    iii.mPlayfairblock = CALL DMPlayfair( block, binkey, Randomkeys(i))
    iv. LET decryptedtext = CONCATENATE( decryptedtext, mPlayfairblock)
    END FOR
11. LET plaintext = encryptedtext
12. OUTPUT plaintext
13. END
```

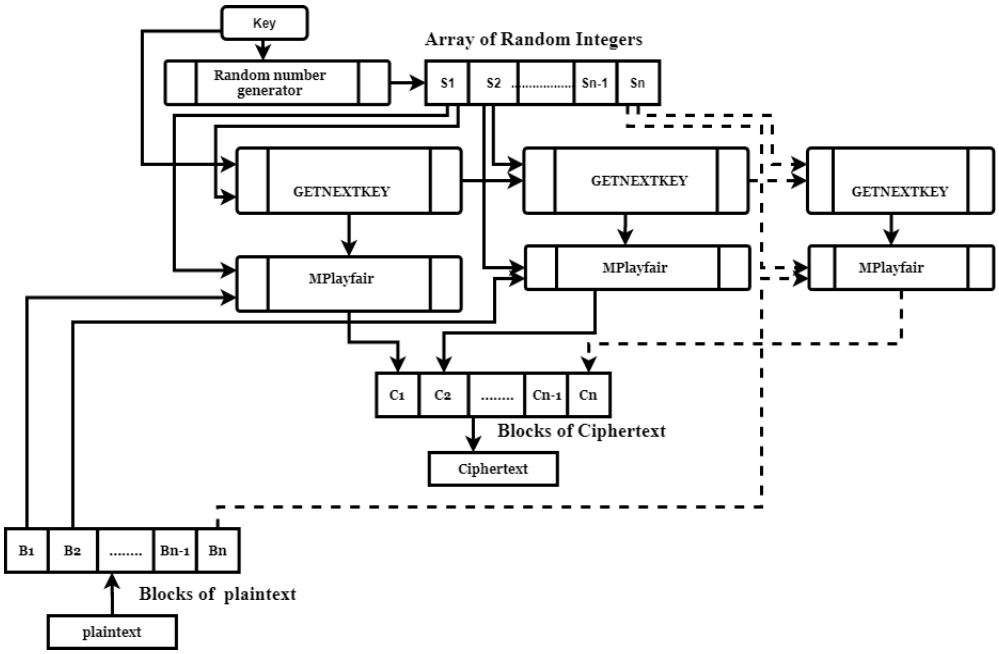


Figure 1. Encryption process of MPF

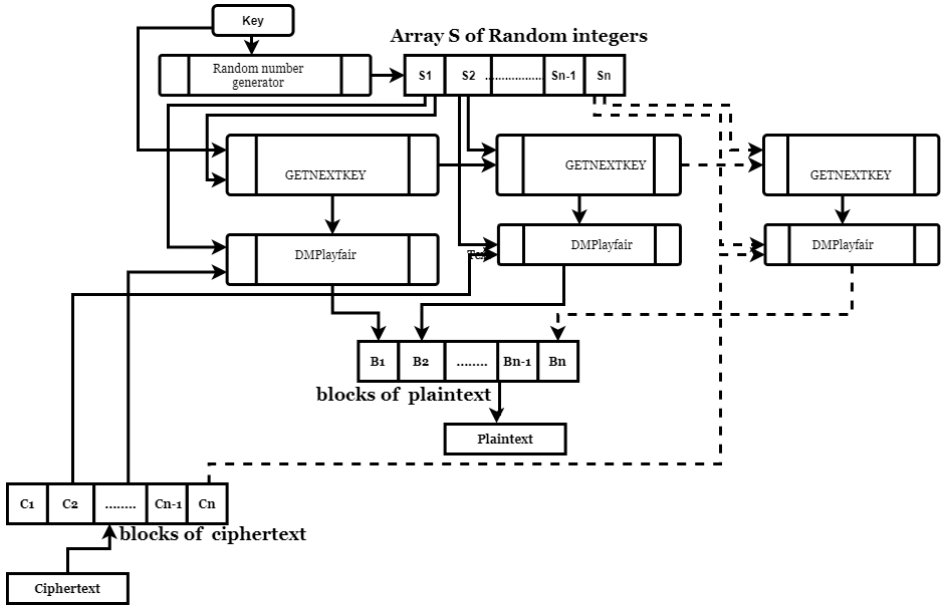


Figure. 2 MPF decryption process

4. RESULTS AND DISCUSSIONS

Analysis of MPF was carried out on Hewlett Packard laptop with AMD E1-1200 APU with Radeon(tm) HD Graphics 1.40 GHz, 4.00 GB (3.59 GB usable), 64-bit Windows 10 operating system, x64-based processor. The Scientific Python Development Environment, Copyright © 2009-2020 Spyder was used throughout the development and analysis

4.1. Encryption

Samples of plaintext and the obtained ciphertexts when the plaintexts are made to pass through the MPF encryption algorithm are given in Table 1. The sample outputs from the encryption function show that the repetitive terms in the plaintext are excluded from the encrypted text and it can never be figured out just from the encrypted text that there was any repetition in the plaintext. Hence, the use of repetitive terms in the plaintext cannot assist cryptanalysts in obtaining any information that can be useful for attacking the MPF cipher presented in this study

Table 1. Sample plaintext and the obtained ciphertext from modified caesar cipher

Plaintext to be encrypted	Encrypted text
wwwwwwwwwwwwwww	w)«#\x11Ùªöï®2B¶aË\x1c
aaaaaaaabbbbbbbbddddwwwwww	\x04\x9au^T\xa0è\x9a'H|ÃÖMî\x98þ×\\[sûtÎÜ!
God is good all the time. great is	Ìã »Kë\x85çè\x01Þc2äÒ\x08\x80*2h\x14´CX\x86VQ\x81Í#_\x8dz
the lord.	\x02'ò©AØ\x98Fn\x17\x1bºÏZ

4.2. Resistance to primitive security attacks

Primitive security attacks include known-plaintext attack (KPA), chosen-plaintext attacks (CPA), chosen-ciphertext attacks (CCA). These attacks are effective if the relation between plaintext and ciphertext is one-to-one [23], [24]. In order to ascertain the resistance of MPF to these primitive attacks, a sample of plaintext that spans 4 blocks is encrypted using MPF. The ciphertext of each block is examined in order to find out if the relationship between plaintext and ciphertext is one-to-one in MPF. The results of the experiment are shown in Table 2.

As can be seen from the resulting block ciphertext in Table 2, though the plaintext is the same for the four blocks different ciphertexts are obtained for each block. These results show that the relationship between the plaintext and ciphertext is not a one-to-one relationship. Hence, MPF is resistant to KPA, CPA, and CCA. This result also implies that a dictionary attack on MPF will do no better than a mere brute force attack. Hence the MPF is resistant to dictionary attack.

Table 2. Relationship between plaintext blocks and ciphertext blocks in MPF

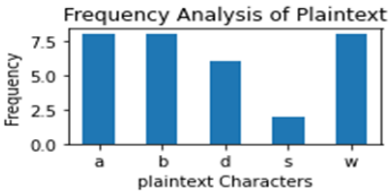
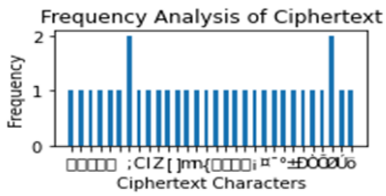
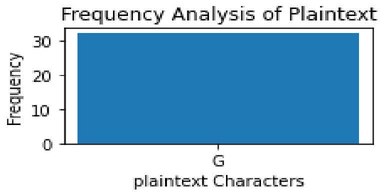
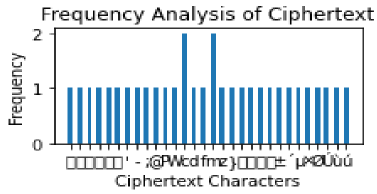
Plaintext: GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG		
Block	Block plaintext	Block ciphertext
1	GGGGGGGGGGGGGGGG	ÀXù\x85v|\x0b\x8d¥|\x92q\x94\x12ÉN
2	GGGGGGGGGGGGGGGG	ó0s\x99\x93Ú0W\x17ië¡pÖÔ\x01
3	GGGGGGGGGGGGGGGG	âÁ\x02½>\x8eè\x95òÖ\x1eëÛó[õ
4	GGGGGGGGGGGGGGGG	c¯\x14U½,\x8a\x8c\x85ÐEhÎ\x8eË\x08

4.3. Spectral frequency analysis of MPF

Frequency analysis is a type of ciphertext attack that is a passive traffic analysis scenario [25] and it can assist the cryptanalyst in getting information about the key, the plaintext, or both. Spectra frequency analysis of MPF is carried out in order to test whether or not it is resistant to frequency analysis attacks. An experiment where different strings were encrypted was set up in order to carry out a spectra frequency analysis of characters of both plaintext and ciphertext. The results of the experiment are shown in Table 3.

Comparing the frequency of the plaintext with that of the ciphertext, it can be said that the frequency of ciphertext letters is uniform while that of plaintext is not uniform. A cipher that produces ciphertext that has a uniform frequency of character is said to be resistant to frequency analysis attacks. Hence, MPF is resistant to frequency analysis attacks.

Table 3. Comparison of frequency analysis of plaintext and ciphertext in MPF

Plaintext	Ciphertext	Spectral analysis of frequency of characters in the string of plaintext and ciphertext
aaaaaaaabbbbbbbbddddwwwwwwddssww	\x1f\x8d\x1b\x12¡{;ØnÕ¯[Ú¤Ø\x11\x9c±°C]I\x8a;\x88mZÐ\x07öÒ	 
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG	ù\x81\x87´c;\x02mmc±fz\x0b-×µ\x96\x15\x03@}ú'P\x84WdØÚ\x08\x1d	 

4.4. Differential cryptanalysis of MPF

The robustness of MPF against differential attacks was tested. Two similar plain texts were encrypted using the same key and the relation between the two ciphertexts was determined. Two metrics net pixel change rate (NPCR) and unified average changing intensity (UACI) were used to determine the robustness of modified Caesar against differential attack. NPCR determines the number of characters that are different between two ciphertexts C1 and C2 from two similar plaintexts. The NPCR for two ciphertexts C1 and C2 of length l was calculated (1).

$$NPCR = \frac{\sum_{i=1}^{l} W(i)}{l} \times 100 \quad (1)$$

where $W(i) = \begin{cases} 0, if\ C1(i) = C2 \\ 1, if\ C1(i) \neq C2 \end{cases}$

The UACI on the other hand represents the intensity difference average between two ciphertexts C1 and C2. Like NPCR, UACI is also calculated in percentage. The UACI of 100% means that both ciphertexts are different in amplitude. The UACI of two ciphertexts C1 and C2 of length l was calculated using (2).

$$UACI = \frac{100}{l \times 256} \sum_{i=1}^{l} |C1 - C2| \quad (2)$$

The two ciphertexts C1 and C2 were generated for the calculation of NPCR and UACI by encrypting a sample of chosen plaintext using one secret key to obtain ciphertext C1. The first symbol of the plaintext was then changed to an arbitrary symbol and the encryption process was repeated to obtain ciphertext C2. Table 4 shows some samples of plaintexts, the generated ciphertexts, and the calculated NPCR and UACI.

A cipher that causes a big change in ciphertext as a result of a little change in plaintext is said to be resistant to differential attacks [2]. Although the NPCR and UACI vary with different keys, the values of NPCR and UACI from the samples shown in Table 4 show that the MPF is resistant to differential attack as 100% and 96.875% of the ciphertext characters changed as a result of a change of one character of the plaintext in sample 1 and 2 respectively. The UACI values obtain also establish that MPF is resistant to differential cryptanalysis.

Table 4. Sample of some calculated NPCR and UACI from encryption in MPF

Plaintext 1	Plaintext 2	C1	C2	NPCR (%)	UACI (%)
GGGGGGGGGG GGGGGG	ÇGGGGGGGGG GGGGGG	]\x9a\xa0]\x9f\x11Lö\x98%ío Ï\x16\x83\x92	\"ºíZð\x80oðÅ8Q7\x1cû¡a\x93	100	33.19
Come and be my friend in deed..	Ãome and be my friend in deed..	\x0e6Ïn\x95L;\x8fu%èÍ\x84D ä3¨hüÄ·\x01òoÆV\rÞÒã²û.	®ø¥\x93'ñ\x8f£\x83 \x8dÍ°úì\x14\x87\x1aZU\x86\x9 c¥ãáÐ+\x93\x8cä%p	96.87 5	31.53

4.5. Information entropy analysis of modified Caesar

Information entropy of a message m encrypted with 2^N possible symbols is calculated by (3),

$$H(m) = \sum_{i=0}^{2^N-1} P(m_i) \log_2 \left(\frac{1}{P(m_i)}\right) \quad (3)$$

where N represents the number of bits of message m, 2^Nrepresents all possible symbols, $P(m_i)$ represents the probability of m_i, $\log_2$ represents the base 2 logarithm and H(m) is the entropy.

In an ideal situation, a message m encrypted with 2^N possible symbols will have entropy $H(m) = N$. MPF uses the whole of 256 ASCII symbols, so the maximum entropy of the encrypted text is expected to be approximately 7.97. A sample text which consists of 219 words, 1, 258 characters (with no space), and 1, 476 (characters with spaces) is encrypted and the entropy of the plaintext and that of the obtained ciphertext were calculated. The obtained results are given in Table 5.

Comparing the calculated entropy of the ciphertext (7.86) with the expected entropy value (7.97), it can be said that MPF has an entropy value very close to the maximum value. This means that the diffusion process of the MPF is strong. A good cipher with efficient diffusion ensures that all the symbols of the plaintext is modified when it is encrypted. An inefficient diffusion stage will make the ciphertext have many identical symbols and therefore make the cipher vulnerable to entropy attack. Hence MPF is resistant to information entropy analysis.

Table 5. Entropy of plaintext and ciphertext

Text type	Entropy
Plaintext	4.25
ciphertext	7.86

4.6. Autocorrelation analysis of modified caesar

Autocorrelation is used for calculating the similarity between two sequences: the plaintext and the plaintext displaced t positions. In classical ciphers, autocorrelation can be used to determine the length of a secret key. If the autocorrelation of the ciphertext is uniform and reduced compared with the autocorrelation of the plaintext, the cipher is said to be resistant to autocorrelation attacks. The graph of the autocorrelation of the text in sub-section 4.5 and the autocorrelation of the obtained ciphertext when MPF is used for encryption is given in Figure 3. The graph in Figure 3 shows that the autocorrelation of the ciphertext is uniform and reduced compared to the autocorrelation of the plaintext. Hence MPF is resistant to autocorrelation cryptanalysis attacks.

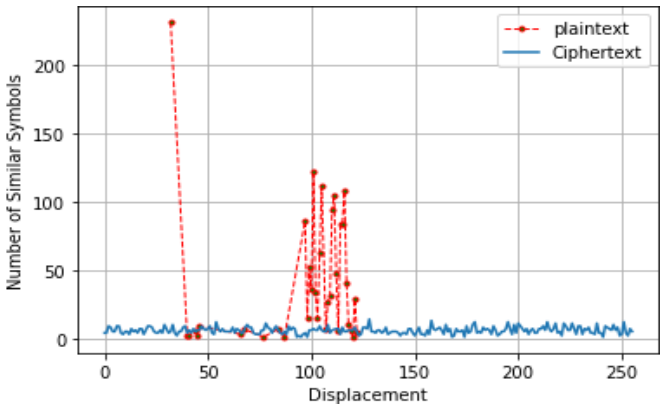


Figure 3. Sample of the graph of autocorrelation of a given plaintext and ciphertext

4.7. Resistance of MPF to brute force attack

The most popular attack used on cryptosystems is the exhaustive search attack or brute-force attack [26]. This attack requires that the attacker tries each possible key until the cryptosystem is broken. Since attackers usually have access to fast computers like supercomputers, the keyspace of a cryptosystem that will resist brute force attacks should be very large. Álvarez and Li [27] suggests that a cryptosystem should have more than 2^{100} keyspace (all strong) for it to be resistant to brute force attack nowadays. Although the keyspace of MPF can be varied, the keyspace is designed to accommodate a large keyspace that is resistant to brute force attacks. Considering the actual technology, the fastest supercomputer today (summit) is capable of 200 PFLOPS (10^{15} floating-point operations per second) or 200,000 trillion calculations per second [3]. It will take this supercomputer 1.99 x 10^{23} years to break a cryptosystem that has the keyspace of 2^{100} according to (4) if it is assumed that the computer is capable of 1000 FLOPS per checking. The least keyspace of MPF is set to 2^{128}.

$$Years = \frac{key\ combination \times 1000}{FLOPS} \times 31536000 \quad (4)$$

4.7.1. Comparison of MPF with existing versions of Playfair

The existing versions of various modifications to Playfair cipher are compared with MPF. The comparisons are based on the number of characters (NC) that are accommodated in each version, the size of the matrix key (MS), and the number of characters that can be handled as the size of the block. (BS). Table 6 summarizes the results of the comparison of MPF with the existing modifications to the Playfair cipher.

Table 6. Comparison of MPF with existing modifications to payfair cipher

Playfair cipher version	NC	MS	BS
[14]	256	16 x 16	2
[9]	28	7 x 4	2
[4]	256	16 x 16	2
[10]	256	Not specified	2
[28]	49	7 x 7	2
[29]	25	5 x 5	2
[22]	36	9 x 4	2
MPF	256	16 X 16	Variable (16 or more)

5. CONCLUSION

Playfair cipher was a powerful cipher in the olden days but it is losing its potency nowadays because of the sophistication in computing devices which possess features that can make them break ciphers such as Playfair within a few seconds. Researchers have worked on and proposed several modified versions of the cipher but attention has been on modification of the key matrix sizes and perhaps different techniques to introduce diffusion and confusion properties into the cipher. However, little or no effort has been made on how the size of the block to be treated as a unit could be increased. A modified version of the Playfair cryptosystem that is capable of handling different block sizes is developed and cryptanalysis was carried out on the cipher. The results from the analysis show that the developed cipher, MPF, is resistant to various cryptanalysis attacks. With this excellent performance, the MPF cryptosystem can be used for securing data in the present age.

REFERENCES

[1] D. Bhowmik, A. Datta, and S. Sinha, "An approach towards analyzing strict key avalanche criterion of block ciphers," International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 11640-11644, 2016 doi: 10.15680/IJIRCCE.2016.040628.

[2] O. E. Omolara, A. I. Oludare, and S. E. Abdulahi, "Developing a modified hybrid caesar cipher and vigenere cipher for secure data communication," *Computer Engineering and Intelligent System*, vol. 5, no. 5, pp. 2222–1719, 2014.

[3] M. Karuppiah, S. Ramanujam, and A. Professor, "Designing an algorithm with high avalanche effect," *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 1, p. 106, 2011.

[4] I. Eweoya, O. Daramola, and N. Omoregbe, "Improving security using refined 16x16 Playfair cipher for enhanced advanced encryption standard (AES)," *Covenant Journal of Informatics and Communication Technology*, vol. 1, no. 2, pp. 79–88, 2013.

[5] R. Anderson, *A guide to building dependable distributed system*. John Wiley & Sons, 2001.

[6] F. Humendru and T. Zebua, "Implementation of triple transposition vegenere cipher algorithm and cipher block chaining for encoding text," *International Journal of Informatics and Computer Science*, vol. 2, no. 1, pp. 26–31, 2018.

[7] U. Thirupalu and E. K. Reddy, "A new cryptosystem for ciphers using transposition techniques," *International Journal of Engineering Research & Technology (IJERT)*, vol. 8, no. 04, pp. 402–406, 2019.

[8] S. G. Srikantaswamy, "Improved caesar cipher with random number generation technique and multistage encryption," *International Journal on Cryptography and Information Security*, vol. 2, no. 4, pp. 39–49, Dec. 2012, doi: 10.5121/ijcis.2012.2405.

[9] [A. A. Alam, B. S. Khalid, and C. M. Salam, "A modified version of Playfair cipher using 7×4 matrix," *International Journal of Computer Theory and Engineering*, pp. 626–628, 2013, doi: 10.7763/ijcte.2013.v5.762.

[10] S. Abdulkhaleq Noaman, "Adaptive Playfair cipher crypto algorithm," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 9, no. 2, 2017, doi: 10.29304/jqcm.2017.9.2.320.

[11] F. I. Lubis, H. F. S. Simbolon, T. P. Batubara, and R. W. Sembiring, "Combination of caesar cipher modification with transposition cipher," *Advances in Science, Technology and Engineering Systems*, vol. 2, no. 5, pp. 22–25, 2017, doi: 10.25046/aj020504.

[12] P. Murali and G. Senthilkumar, "Modified version of Playfair cipher using linear feedback shift register," *Proceedings - 2009 International Conference on Information Management and Engineering, ICIME 2009*, 2009, pp. 488–490, doi: 10.1109/ICIME.2009.86.

[13] D. A. Negi, "Cryptography Playfair cipher using linear feedback shift register," *IOSR Journal of Engineering*, vol. 02, no. 05, pp. 1212–1216, 2012, doi: 10.9790/3021-020512121216.

[14] S. Hamad, A. Khalifa, A. Elhadad, and S. Rida, "A modified Playfair cipher for encrypting digital images," *Journal of Communications and Computer Engineering*, vol. 3, no. 2, p. 1, 2014, doi: 10.20454/jcce.2013.731.

[15] S. S. Srivastava and N. Gupta, "Optimization and analysis of the extended Playfair cipher," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, Apr. 2011, pp. 267–270, doi: 10.1109/ETNCC.2011.6255901.

[16] S. S. Dhenakaran and M. Ilayaraja, "Extension of Playfair cipher using 16x16 matrix," *International Journal of Computer Applications*, vol. 48, no. 7, pp. 37–41, Jun. 2012, doi: 10.5120/7363-0192.

[17] M. J. Cowan, "Breaking short Playfair ciphers with the simulated annealing algorithm," *Cryptologia*, vol. 32, no. 1, pp. 71–83, Jan. 2008, doi: 10.1080/01611190701743658.

[18] M. Cowan, "Churn Algorithm," 2015. https://web.archive.org/web/20141129010802/http://www.cryptoden.com/index.php/algorithms/churn-algorithm/20-churn-algorithm, (accessed Apr. 14, 2022).

[19] N. R. Alkazaz, S. A. Irvine, and W. J. Teahan, "An automatic cryptanalysis of simple substitution ciphers using compression," in *Proceedings of the 1st Conference on Historical Cryptology*, Jan. 2018, pp. 115– 124.

[20] R. K. Babu, S. U. Kumar, A. V. Babu, I. V. N. Aditya, and P. Komuraiah, "An extension to traditional Playfair cryptographic method," *International Journal of Computer Applications*, vol. 17, no. 5, pp. 34–36, Mar. 2011, doi: 10.5120/2213-2814.

[21] M. Sabry, M. Hashem, N. Taymoor, and M. E. Khalifa, "A DNA and amino acids-based implementation of Playfair cipher," *International Journal of Computer Science and Information Security*, vol. 8, no. Jan 2010, pp. 19219–19226, 2010.

[22] R. S. Ali, R. K. Hassoun, I. F. Jaleel, and N. S. Ali, "Proposal for encryption by using modified play fair algorithm and bioinformatics techniques," in *ACM International Conference Proceeding Series*, 2019, pp. 120–126, doi: 10.1145/3321289.3321321.

[23] A. Sengupta and U. K. Ray, "Message mapping and reverse mapping in elliptic curve cryptosystem," *Security and Communication Networks*, vol. 9, no. 18, pp. 5363–5375, 2016, doi: 10.1002/sec.1702.

[24] R. Lewand, *Cryptological Mathematics*. Providence, Rhode Island: American Mathematical Society, 2000.

[25] B. Padma, D. Chandravathi, and L. Pratibha, "Defense against frequency analysis in elliptic curve cryptography using k-means clustering," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Feb. 2021, pp. 64–69, doi: 10.1109/ICCCIS51004.2021.9397065.

[26] T. Mekhaznia and A. Zidani, "Swarm intelligence algorithms in cryptanalysis of simple Feistel ciphers," *International Journal of Information and Communication Technology*, vol. 13, no. 1, p. 114, 2018, doi: 10.1504/IJICT.2018.090436.
[27] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *International Journal of Bifurcation and Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006, doi: 10.1142/S0218127406015970.
[28] M. A. T. Shakil and M. R. Islam, "An efficient modification to Playfair cipher," *ULAB Journal of Science and Engineering*, vol. 5, no. 1, pp. 26–30, 2014.
[29] [M. V. Ahamad, M. U. Siddiqui, M. Masroor, and U. Fatima, "An improved Playfair encryption technique using Fibonacci series generated secret key," *International Journal of Engineering and Technology (UAE)*, vol. 7, no. 4, pp. 347–351, 2018, doi: 10.14419/ijet.v7i4.5.20104.

BIOGRAPHIES OF AUTHORS

Esau Taiwo Oladipupo received his first degree (B. Tech) Computer Engineering from the Department of Computer Science and Engineering, Ladoke Akintola University of Technology in 2003. He has also Master degree (M. Tech) Computer Science, from the same institution in 2016. He has been a Lecturer in the Department of Computer Science the Federal Polytechnic Bida since 2009. He is currently pursuing his Ph.D. in Computer Science from the University of Ilorin, Nigeria. His research area includes Computer and Communication Security, Crytography and Biometric. He can be contacted at email: taiwotheophilus@gmail.com or oladipupo.esau@fedpolybida.edu.ng

Oluwakemi Christiana Abikoye has a B.Sc, M.Sc. and Ph.D degrees in Computer Science. She is an academic staff at the Department of Computer Science, Faculty of Communication and Information Sciences, University of Ilorin, Ilorin, Nigeria. Her research interests include Cryptography, Computer and Communication Network (Cyber) Security, Biometrics, Human Computer Interaction and Text and Data Mining. She can be contacted at email: abikoye.o@unilorin.edu.ng