A Novel Square-Expanded-Matrix-Rotation (SEMR) Cryptography Method

Suyash Kandele, Veena Anand
Department of Computer Science and Engineering
National Institute of Technology Raipur, India

***Abstract*—The proposed algorithm is a symmetric algorithm. It employs a key of 8-bit. The algorithm focuses on breaking the input string into a large number of small sized square matrices, whose size varies from 1x1 to 9x9. On each square matrix, we first apply displacement method, which changes the position of characters in the string, and then add expanded matrix, which hides the number of occurrences and values of characters. In each iteration, the value of key gets updated on the basis of the characters encountered in the string encrypted so far. Thus, the key becomes more complicated after every step, thereby increasing the strength of encryption and also making its decryption more difficult. The diverse operations performed on different parts of the string makes it excessively complicated. The proposed algorithm is highly unpredictable and, therefore, changes dynamically with the variation in string length and string characters.**

Keywords—Square Matrix; Matrix Manipulation; Expanded Matrix; Remainder Processing; Rotation Operation;

I. INTRODUCTION

With the immense development in the usage and users of Internet over the two decades, the security of data has emerged as a crucial aspect along with increasing the efficiency. The data cannot be sent on a shared medium without the covering of tough cryptography system. The development of new techniques is unable to surpass the rate of attack on the already existing systems. Thus, there is a call for the development of highly complex and fickle mechanism that could change on its own to provide enhanced protection to the priceless data.

In the present work, the author has used several methodology derived from the combination of numerous basic operations and functions. The focus is to reduce the chances of anticipation by the intruders; who are growing in numbers and technology; by changing the structure of statement and the sequence of the elements, and varying the frequency and value of characters. The method is divided into several iterations and the key used here updates itself to form a more complex key after every iteration.

II. BASIC TERMINOLOGY

A. Square Matrix

Square Matrix is a 2-dimensional array that has same number of columns as the number of rows. Its size is denoted by NxN where, N is number of rows as well as number of columns.

B. Matrix Manipluation

Matrix Manipulation comprises of a series of operations performed on square matrix to modify it.

C. Magic Matrix

Magic Matrix is a square matrix possessing the special property in which the elements are arranged in such a way that the sum of elements of each column, of each row and of the two diagonals is equal.

D. Expanded Matrix

Expanded Matrix of size NxN is a special square matrix created in this method from magic matrix of size (N-2)x(N-2) by performing some shift operations on magic matrix and assigning some calculated value to the new positions introduced during shifting.

E. Remainder Processing

Remainder Processing constitutes of a series of operations performed on the remainder elements.

F. XOR Operation

Here, the bitwise XOR operation is performed on various numbers. When the two bits are identical, the result is evaluated to zero, otherwise to one.

G. Left Rotation Operation

Here, the Left Rotation operation is performed on the 8-bit numbers. Left Rotation by 1-bit causes the MSB (Most Significant Bit) to be shifted to LSB (Least Significant Bit) and all other bits to be shifted to 1-position to the left, i.e. towards MSB.

H. Right Rotation Operation

Here, the Right Rotation operation is performed on the 8-bit numbers. Right Rotation by 1-bit causes the LSB (Least Significant Bit) to be shifted to MSB (Most Significant Bit) and all other bits to be shifted to 1-position to the right, i.e. towards LSB.

III. PROPOSED ENCRYPTION ALGORITHM

Step-1
Input the plain text PLAIN_TEXT and the key, KEY.

Step-2
Convert each element of the input string PLAIN_TEXT into its corresponding ASCII value and calculate its length, LEN.

Step-3

Set the values:

i. REM_LEN = LEN
ii. PREV_GEN = KEY

Step-4

If REM_LEN>81, then
    Goto Step-5.
Else
    If REM_LEN>7, then
        Goto Step-6.
    Else
        Goto Step-8.

Step-5

Perform following operations.

i. Calculate:
   a. S = sum of digits in REM_LEN
   b. M = smallest digit in REM_LEN greater than 0
   c. N = S + M
ii. Convert N into a single digit number.
iii. Goto Step-7.

Step-6

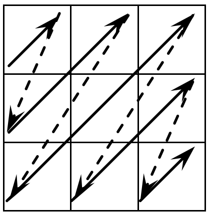
Calculate:

N = floor (square_root (REM_LEN / 2))

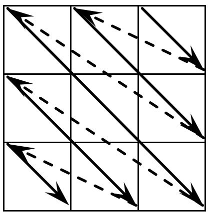
Step-7

Perform Matrix Manipulation.

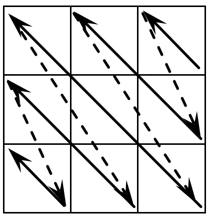
i. Store the value of N in the array MAT_SIZE.
ii. Extract (N*N) values from PLAIN_TEXT and store them diagonally-upward-left-to-right from top-left corner to right-bottom corner in the square matrix TEMP_MAT.



iii. Calculate NEXT_GEN by performing XOR between all the elements of TEMP_MAT.
iv. Perform XOR on calculated NEXT_GEN with KEY.
v. Perform XOR on all elements in the matrix TEMP_MAT with PREV_GEN.
vi. Set PREV_GEN = NEXT_GEN.
vii. If the N is Odd, then
        Read the elements of TEMP_MAT diagonally -downward-left-to-right from top-right corner to left-bottom corner and store in an array TEMP_ARR.



    Else
        Read the elements of TEMP_MAT diagonally –upward-right-to-left from top-right corner to left-bottom corner and store in an array TEMP_ARR.

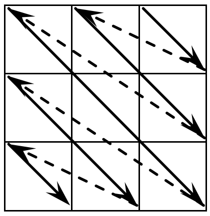


viii. Create Expanded Matrix EXP_MAT of size NxN by following steps:
   a. Create a magic matrix of size (N–2)x(N–2).
   b. Shift the elements below the auxiliary diagonal of the magic matrix by one position to downward direction and by one position to right direction and store into EXP_MAT. Set the value of newly introduced positions to zero.
   c. Shift the elements below the main diagonal of the EXP_MAT by one position to downward direction and the elements above the main diagonal by one position to right direction and store into EXP_MAT. Set the value of newly introduced positions to zero.
   d. For each element a[i][j] in the matrix EXP_MAT that has its value zero, assign it the value $(i^2 + j^3)$.
ix. Read the Expanded Matrix EXP_MAT in row major order and store in 1-dimensional array EXP_ARR.
x. Add the corresponding elements of EXP_ARR to TEMP_ARR.
xi. If the value of element of EXP_ARR is Even
        Left rotate the element of TEMP_ARR by 1-bit.
    Else
        Right rotate the element of TEMP_ARR by 1-bit.
xii. Append the array TEMP_ARR at the end of cipher text CIPHER_TEXT.
xiii. Set REM_LEN = REM_LEN – (N * N)
xiv. Goto Step-4.

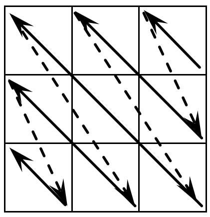
Step-8

Perform Remainder Manipulation.

i. Create a magic matrix MAG_MAT of size 3x3.
ii. If the number of remainder elements is Odd, then
   a. Read the elements of magic matrix MAG_MAT diagonally –downward-left-to-right from top-right corner to left-bottom corner and store in 1-dimensional array MAG_ARR.



   b. If element of MAG_ARR is Odd, then
          Store the square of the element in MAG_ARR.
      Else,
          Store the cube of element.
   Else,
   a. Read the elements of magic matrix MAG_MAT diagonally– upward – right – to – left from top-right corner to left-bottom corner and store in 1-dimensional array MAG_ARR.



   b. If element of MAG_ARR is Even, then

Store the square of the element in MAG_ARR.

Else,

Store the cube of element.

iii. Perform XOR operation between the remainder elements REM and MAG_ARR and store the result in REM.

iv. If the element in REM is at even position, then

Right rotate the element by (8–position) bits.

Else

Left rotate the element by (8–position) bits.

v. Perform XOR operation on REM with KEY.

vi. Append the array REM at the end of cipher text CIPHER_TEXT.

Step-9

Print the CIPHER_TEXT.

IV. EXAMPLE OF ENCRYPTION ALGORITHM

Step-1

Consider the entered PLAIN_TEXT is :

> This is a sample string, which is being used to test the results and efficiency of an Cryptography Algorithm.

KEY is 77.

Step-2

Here, the ASCII equivalent of the PLAIN_TEXT is [84 104 105 115 32 105 115 32 97 32 115 97 109 112 108 101 32 115 116 114 105 110 103 44 32 119 104 105 99 104 32 105 115 32 98 101 105 110 103 32 117 115 101 100 32 116 111 32 116 101 115 116 32 116 104 101 32 114 101 115 117 108 116 115 32 97 110 100 32 101 102 102 105 99 105 101 110 99 121 32 111 102 32 97 110 32 67 114 121 112 116 111 103 114 97 112 104 121 32 65 108 103 111 114 105 116 104 109 46]

Length of string, LEN = 109

Step-3

In this example,

REM_LEN = 109

PREV_GEN = 77

1st Iteration

Step-4

REM_LEN = 109

REM_LEN>81 : TRUE

Goto Step-5

Step-5

S = 1 + 0 + 9 = 10

M = 1

N = 10 + 1 = 11

N = 1 + 1 = 2

Step-7

i. MAT_SIZE = [2]

ii. TEMP_ARR = [84 104 105 115]

$$TEMP\_MAT = \begin{bmatrix} 84 & 105 \\ 104 & 115 \end{bmatrix}$$

iii. NEXT_GEN = 38

iv. NEXT_GEN = 107

v. PREV_GEN = 77

$$TEMP\_MAT = \begin{bmatrix} 25 & 36 \\ 37 & 62 \end{bmatrix}$$

vi. PREV_GEN = 107

vii. TEMP_ARR = [36 62 25 37]

viii.

BASE = [0]

$$EXP\_MAT = \begin{bmatrix} 2 & 9 \\ 5 & 12 \end{bmatrix}$$

ix. EXP_ARR = [2 9 5 12]

x. TEMP_ARR = [38 71 30 49]

xi. TEMP_ARR = [76 163 15 98]

xii. CIPHER_TEXT = [76 163 15 98]

xiii. REM_LEN = 105

xiv. Goto Step-4.

2nd Iteration

Step-4

REM_LEN = 105

REM_LEN>81 : TRUE

Goto Step-5

Step-5

S = 1 + 0 + 5 = 6

M = 1

N = 6 + 1 = 7

Step-7

i. MAT_SIZE = [2 7]

ii. TEMP_ARR = [32 105 115 32 97 32 115 97 109 112 108 101 32 115 116 114 105 110 103 44 32 119 104 105 99 104 32 105 115 32 98 101 105 110 103 32 117 115 101 100 32 116 111 32 116 101 115 116 32]

$$TEMP\_MAT = \begin{bmatrix} 32 & 115 & 32 & 112 & 116 & 32 & 105 \\ 105 & 97 & 109 & 115 & 44 & 32 & 110 \\ 32 & 97 & 32 & 103 & 104 & 105 & 101 \\ 115 & 101 & 110 & 99 & 101 & 115 & 111 \\ 108 & 105 & 105 & 98 & 117 & 116 & 101 \\ 114 & 104 & 32 & 32 & 32 & 116 & 116 \\ 119 & 115 & 103 & 100 & 32 & 115 & 32 \end{bmatrix}$$

iii. NEXT_GEN = 110

iv. NEXT_GEN = 35

v. PREV_GEN = 107

$$TEMP\_MAT = \begin{bmatrix} 75 & 24 & 75 & 27 & 31 & 75 & 2 \\ 2 & 10 & 6 & 24 & 71 & 75 & 5 \\ 75 & 10 & 75 & 12 & 3 & 2 & 14 \\ 24 & 14 & 5 & 8 & 14 & 24 & 4 \\ 7 & 2 & 2 & 9 & 30 & 31 & 14 \\ 25 & 3 & 75 & 75 & 75 & 31 & 31 \\ 28 & 24 & 12 & 15 & 75 & 24 & 75 \end{bmatrix}$$

vi. PREV_GEN = 35

vii. TEMP_ARR = [2 75 5 31 75 14 27 71 2 4 75 24 3 24 14 24 6 12 14 31 31 75 10 75 8 30 31 75 2 10 5 9 75 24 75 14 2 75 75 24 2 75 15 7 3 12 25 24 28]

viii.

$$BASE = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

$$EXP\_MAT = \begin{bmatrix} 2 & 17 & 24 & 1 & 8 & 15 & 344 \\ 17 & 12 & 5 & 7 & 14 & 220 & 15 \\ 23 & 5 & 36 & 13 & 134 & 14 & 16 \\ 4 & 6 & 13 & 80 & 13 & 20 & 22 \\ 10 & 12 & 52 & 13 & 150 & 21 & 3 \\ 11 & 44 & 12 & 19 & 21 & 252 & 9 \\ 50 & 11 & 18 & 25 & 2 & 9 & 392 \end{bmatrix}$$

ix. EXP_ARR = [2 17 24 1 8 15 344 17 12 5 7 14 220 15 23 5 36 13 134 14 16 4 6 13 80 13 20 22 10 12 52 13 150 21 3 11 44 12 19 21 252 9 50 11 18 25 2 9 392]

x. TEMP_ARR = [4 92 29 32 83 29 115 88 14 9 82 38 223 39 37 29 42 25 148 45 47 79 16 88 88 43 51 97 12 22 57 22 225 45 78 25 46 87 94 45 254 84 65 18 21 37 27 33 164]

xi. TEMP_ARR = [8 46 58 16 166 142 230 44 28 132 41 76 191 147 146 142 84 140 41 90 94 158 32 44 176 149 102 194 24 44 114 11 195 150 39 140 92 174 47 150 253 42 130 9 42 146 54 144 73]

xii. CIPHER_TEXT = [76 163 15 98 8 46 58 16 166 142 230 44 28 132 41 76 191 147 146 142 84 140 41 90 94 158 32 44 176 149 102 194 24 44 114 11 195 150 39 140 92 174 47 150 253 42 130 9 42 146 54 144 73]

xiii. REM_LEN = 56

xiv. Goto Step-4.

3rd Iteration

Step-4

REM_LEN = 56
REM_LEN>81 : FALSE
REM_LEN>7 :TRUE
Goto Step-6

Step-6

N = floor (square_root (56 / 2))
= floor (square_root (28))
= floor (5.291)
= 5

Step-7

i. MAT_SIZE = [2 7 5]

ii. TEMP_ARR = [116 104 101 32 114 101 115 117 108 116 115 32 97 110 100 32 101 102 102 105 99 105 101 110 99]

$$TEMP\_MAT = \begin{bmatrix} 116 & 101 & 101 & 116 & 100 \\ 104 & 114 & 108 & 110 & 102 \\ 32 & 117 & 97 & 102 & 105 \\ 115 & 32 & 101 & 99 & 110 \\ 115 & 32 & 105 & 101 & 99 \end{bmatrix}$$

iii. NEXT_GEN = 38

iv. NEXT_GEN = 107

v. PREV_GEN = 35

$$TEMP\_MAT = \begin{bmatrix} 87 & 70 & 70 & 87 & 71 \\ 75 & 81 & 79 & 77 & 69 \\ 3 & 86 & 66 & 69 & 74 \\ 80 & 3 & 70 & 64 & 77 \\ 80 & 3 & 74 & 70 & 64 \end{bmatrix}$$

vi. PREV_GEN = 107

vii. TEMP_ARR = [71 87 69 70 77 74 70 79 69 77 87 81 66 64 64 75 86 70 70 3 3 74 80 3 80]

viii.

$$BASE = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

$$EXP\_MAT = \begin{bmatrix} 2 & 8 & 1 & 6 & 126 \\ 8 & 12 & 5 & 68 & 6 \\ 3 & 5 & 36 & 5 & 7 \\ 4 & 24 & 5 & 80 & 2 \\ 26 & 4 & 9 & 2 & 150 \end{bmatrix}$$

ix. EXP_ARR = [2 8 1 6 126 8 12 5 68 6 3 5 36 5 7 4 24 5 80 2 26 4 9 2 150]

x. TEMP_ARR = [73 95 70 76 203 82 82 84 137 83 90 86 102 69 71 79 110 75 150 5 29 78 89 5 230]

xi. TEMP_ARR = [146 190 35 152 151 164 164 42 19 166 45 43 204 162 163 158 220 165 45 10 58 156 172 10 205]

xii. CIPHER_TEXT = [76 163 15 98 8 46 58 16 166 142 230 44 28 132 41 76 191 147 146 142 84 140 41 90 94 158 32 44 176 149 102 194 24 44 114 11 195 150 39 140 92 174 47 150 253 42 130 9 42 146 54 144 73 146 190 35 152 151 164 164 42 19 166 45 43 204 162 163 158 220 165 45 10 58 156 172 10 205]

xiii. REM_LEN = 31

xiv. Goto Step-4.

4th Iteration

Step-4

REM_LEN = 31
REM_LEN>81 : FALSE
REM_LEN>7 :TRUE
Goto Step-6

Step-6
N = floor (square_root (31 / 2))
= floor (square_root (15.5))
= floor (3.937)
= 3

Step-7
i. MAT_SIZE = [2 7 5 3]
ii. TEMP_ARR = [121 32 111 102 32 97 110 32 67]

$$TEMP\_MAT = \begin{bmatrix} 121 & 111 & 97 \\ 32 & 32 & 32 \\ 102 & 110 & 67 \end{bmatrix}$$

iii. NEXT_GEN = 28
iv. NEXT_GEN = 81
v. PREV_GEN = 107

$$TEMP\_MAT = \begin{bmatrix} 18 & 4 & 10 \\ 75 & 75 & 75 \\ 13 & 5 & 40 \end{bmatrix}$$

vi. PREV_GEN = 81
vii. TEMP_ARR = [10 4 75 18 75 40 75 5 13]
viii.
BASE = [0]

$$EXP\_MAT = \begin{bmatrix} 2 & 9 & 28 \\ 5 & 12 & 31 \\ 10 & 17 & 36 \end{bmatrix}$$

ix. EXP_ARR = [2 9 28 5 12 31 10 17 36]
x. TEMP_ARR = [12 13 103 23 87 71 85 22 49]
xi. TEMP_ARR = [24 134 206 139 174 163 170 11 98]
xii. CIPHER_TEXT = [76 163 15 98 8 46 58 16 166 142 230 44 28 132 41 76 191 147 146 142 84 140 41 90 94 158 32 44 176 149 102 194 24 44 114 11 195 150 39 140 92 174 47 150 253 42 130 9 42 146 54 144 73 146 190 35 152 151 164 164 42 19 166 45 43 204 162 163 158 220 165 45 10 58 156 172 10 205 24 134 206 139 174 163 170 11 98]
xiii. REM_LEN = 22
xiv. Goto Step-4.

5th Iteration

Step-4
REM_LEN = 22
REM_LEN>81 : FALSE
REM_LEN>7 :TRUE
Goto Step-6

Step-6
N = floor (square_root (22 / 2))
= floor (square_root (11))
= floor (3.317)
= 3

Step-7
i. MAT_SIZE = [2 7 5 3 3]
ii. TEMP_ARR = [114 121 112 116 111 103 114 97 112]

$$TEMP\_MAT = \begin{bmatrix} 114 & 112 & 103 \\ 121 & 111 & 97 \\ 116 & 114 & 112 \end{bmatrix}$$

iii. NEXT_GEN = 100
iv. NEXT_GEN = 41
v. PREV_GEN = 81

$$TEMP\_MAT = \begin{bmatrix} 35 & 33 & 54 \\ 40 & 62 & 48 \\ 37 & 35 & 33 \end{bmatrix}$$

vi. PREV_GEN = 41
vii. TEMP_ARR = [54 33 48 35 62 33 40 35 37]
viii.
BASE = [0]

$$EXP\_MAT = \begin{bmatrix} 2 & 9 & 28 \\ 5 & 12 & 31 \\ 10 & 17 & 36 \end{bmatrix}$$

ix. EXP_ARR = [2 9 28 5 12 31 10 17 36]
x. TEMP_ARR = [56 42 76 40 74 64 50 52 73]
xi. TEMP_ARR = [112 21 152 20 148 32 100 26 146]
xii. CIPHER_TEXT = [76 163 15 98 8 46 58 16 166 142 230 44 28 132 41 76 191 147 146 142 84 140 41 90 94 158 32 44 176 149 102 194 24 44 114 11 195 150 39 140 92 174 47 150 253 42 130 9 42 146 54 144 73 146 190 35 152 151 164 164 42 19 166 45 43 204 162 163 158 220 165 45 10 58 156 172 10 205 24 134 206 139 174 163 170 11 98 112 21 152 20 148 32 100 26 146]
xiii. REM_LEN = 13
xiv. Goto Step-4.

6th Iteration

Step-4
REM_LEN = 13
REM_LEN>81 : FALSE
REM_LEN>7 :TRUE
Goto Step-6

Step-6
N = floor (square_root (13 / 2))
= floor (square_root (6.5))
= floor (2.549)
= 2

Step-7
i. MAT_SIZE = [2 7 5 3 3 2]
ii. TEMP_ARR = [104 121 32 65]

$$TEMP\_MAT = \begin{bmatrix} 104 & 32 \\ 121 & 65 \end{bmatrix}$$

iii. NEXT_GEN = 112
iv. NEXT_GEN = 61
v. PREV_GEN = 41

$$TEMP\_MAT = \begin{bmatrix} 65 & 9 \\ 80 & 104 \end{bmatrix}$$

vi. PREV_GEN = 61

vii. TEMP_ARR = [9 104 65 80]

viii.

BASE = [0]

$$EXP\_MAT = \begin{bmatrix} 2 & 9 \\ 5 & 12 \end{bmatrix}$$

ix. EXP_ARR = [2 9 5 12]
x. TEMP_ARR = [11 113 70 92]
xi. TEMP_ARR = [22 184 35 184]
xii. CIPHER_TEXT = [76 163 15 98 8 46 58
16 166 142 230 44 28 132 41 76 191
147 146 142 84 140 41 90 94 158 32
44 176 149 102 194 24 44 114 11 195
150 39 140 92 174 47 150 253 42 130
9 42 146 54 144 73 146 190 35 152
151 164 164 42 19 166 45 43 204 162
163 158 220 165 45 10 58 156 172 10
205 24 134 206 139 174 163 170 11 98
112 21 152 20 148 32 100 26 146 22
184 35 184]
xiii. REM_LEN = 9
xiv. Goto Step-4.

7th Iteration

Step-4
REM_LEN = 9
REM_LEN>81 : FALSE
REM_LEN>7 :TRUE
Goto Step-6

Step-6
N = floor (square_root (9 / 2))
= floor (square_root (4.5))
= floor (2.121)
= 2

Step-7
i. MAT_SIZE = [2 7 5 3 3 2 2]
ii. TEMP_ARR = [108 103 111 114]

$$TEMP\_MAT = \begin{bmatrix} 108 & 111 \\ 103 & 114 \end{bmatrix}$$

iii. NEXT_GEN = 22
iv. NEXT_GEN = 91
v. PREV_GEN = 61

$$TEMP\_MAT = \begin{bmatrix} 81 & 82 \\ 90 & 79 \end{bmatrix}$$

vi. PREV_GEN = 91
vii. TEMP_ARR = [82 79 81 90]

viii.

BASE = [0]

$$EXP\_MAT = \begin{bmatrix} 2 & 9 \\ 5 & 12 \end{bmatrix}$$

ix. EXP_ARR = [2 9 5 12]
x. TEMP_ARR = [84 88 86 102]
xi. TEMP_ARR = [168 44 43 204]
xii. CIPHER_TEXT = [76 163 15 98 8 46 58
16 166 142 230 44 28 132 41 76 191
147 146 142 84 140 41 90 94 158 32
44 176 149 102 194 24 44 114 11 195
150 39 140 92 174 47 150 253 42 130
9 42 146 54 144 73 146 190 35 152
151 164 164 42 19 166 45 43 204 162
163 158 220 165 45 10 58 156 172 10
205 24 134 206 139 174 163 170 11 98
112 21 152 20 148 32 100 26 146 22
184 35 184 168 44 43 204]
xiii. REM_LEN = 5
xiv. Goto Step-4.

8th Iteration

Step-4
REM_LEN = 5
REM_LEN>81 : FALSE
REM_LEN>7 : FALSE
Goto Step-8

Step-8
REM = [105 116 104 109 46]

i.

$$MAG\_MAT(3x3) = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

ii. MAG_ARR[216 1 49 0 25 2 3 9 4]
iii. REM = [177 117 89 109 55]
iv. REM = [216 213 43 214 185]
v. REM = [149 152 102 155 244]
vi. CIPHER_TEXT = [76 163 15 98 8 46 58
16 166 142 230 44 28 132 41 76 191
147 146 142 84 140 41 90 94 158 32
44 176 149 102 194 24 44 114 11 195
150 39 140 92 174 47 150 253 42 130
9 42 146 54 144 73 146 190 35 152
151 164 164 42 19 166 45 43 204 162
163 158 220 165 45 10 58 156 172 10
205 24 134 206 139 174 163 170 11 98
112 21 152 20 148 32 100 26 146 22
184 35 184 168 44 43 204 149 152 102
155 244]

Step-9
The CIPHER_TEXT is

L£.:¦æ,)L¿T)Z^,°fÂ,r
Ã'\®/ý* *6 • I¾#¤¤*¦-+Ì¢£Ü¥-
:¬
ÍÎ®£ª
bpd¸#¸¨,+Ìfô

V. PROPOSED DECRYPTION ALGORITHM

Step-1
Input the cipher text CIPHER_TEXT and the key, KEY.

Step-2
Convert each element of CIPHER_TEXT into its corresponding ASCII value and calculate its length, LEN.

Step-3
Set the values:
i. REM_LEN = LEN
ii. PREV_GEN = KEY

Step-4

If REM_LEN>81, then
    Goto Step-5.
Else
    If REM_LEN>7, then
        Goto Step-6.
    Else
        Goto Step-8.

Step-5

Perform following operations.

i. Calculate:
    a. S = sum of digits in REM_LEN
    b. M = smallest digit in REM_LEN greater than 0
    c. N = S + M
ii. Convert N into a single digit number.
iii. Goto Step-7.

Step-6

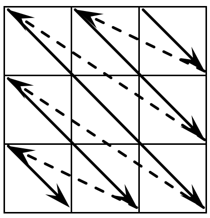
Calculate,

N = floor (square_root (REM_LEN / 2))

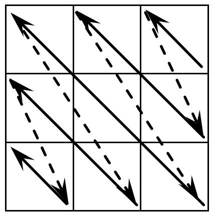
Step-7

Perform Matrix Manipulation.

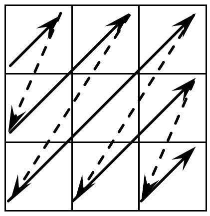
i. Store the value of N in an array MAT_SIZE.
ii. Extract (N*N) values from CIPHER_TEXT and store them in array TEMP_ARR.
iii. Create Expanded Matrix EXP_MAT of size NxN by following steps:
    a. Create a magic matrix of size (N–2)x(N–2).
    b. Shift the elements below the auxiliary diagonal of the magic matrix by one position to downward direction and by one position to right direction and store into EXP_MAT. Set the value of newly introduced positions to zero.
    c. Shift the elements below the main diagonal of the EXP_MAT by one position to downward direction and the elements above the main diagonal by one position to right direction. Set the value of newly introduced positions to zero.
    d. For each element a[i][j] in the matrix EXP_MAT that has its value zero, assign it the value $(i^2 + j^3)$.
iv. Read the Expanded Matrix EXP_MAT in row major order and store in 1-dimensional array EXP_ARR.
v. If the value of element of EXP_ARR is Even, then
        Right rotate the element of TEMP_ARR by 1-bit.
    Else
        Left rotate the element of TEMP_ARR by 1-bit.
vi. Subtract the corresponding elements of EXP_ARR from TEMP_ARR.
vii. If the N is Odd, then
        Read the elements of TEMP_ARR and store into TEMP_MAT diagonally-downward-left-to-right from top-right corner to left-bottom corner.



    Else
        Read the elements of TEMP_ARR and store into TEMP_MAT diagonally-upward-right-to-left from top-right corner to left-bottom corner.



viii. Perform XOR on all elements in the matrix TEMP_MAT with PREV_GEN.
ix. Calculate NEXT_GEN by performing XOR between all the elements of TEMP_MAT.
x. Perform XOR on calculated NEXT_GEN with KEY.
xi. Read the square matrix TEMP_MAT diagonally-upward-left-to-right from top-left corner to right-bottom corner and store in array TEMP_ARR.

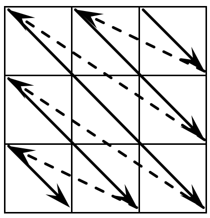


xii. Set PREV_GEN = NEXT_GEN.
xiii. Append the array TEMP_ARR at the end of plain text PLAIN_TEXT.
xiv. Set REM_LEN = REM_LEN – (N * N)
xv. Goto Step-4.

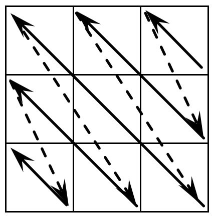
Step-8

Perform Remainder Manipulation

i. Perform XOR operation on REM with KEY.
ii. If the element in REM is at even position, then
        Left rotate the element by (8–position) bits.
    Else
        Right rotate the element by (8–position) bits.
iii. Create a magic matrix MAG_MAT of size 3x3.
iv. If the number of remainder elements is Odd, then
    a. Read the elements of magic matrix MAG_MAT diagonally –downward-left-to-right from top-right corner to left-bottom corner and store in 1-dimensional array MAG_ARR.



    b. If element of MAG_ARR is Odd, then
            Store the square of the element in MAG_ARR.
        Else,
            Store the cube of element.
    Else,
    a. Read the elements of magic matrix MAG_MAT diagonally –upwards-right-to-left from top-right corner to left-bottom corner and store in 1-dimensional array MAG_ARR.



b. If element of MAG_ARR is Even, then
 Store the square of the element in MAG_ARR.
 Else,
 Store the cube of element.

v. Perform XOR operation between the remainder elements REM and MAG_ARR and store the result in REM.
vi. Append the array REM at the end of plain text PLAIN_TEXT.

Step-9
Print the PLAIN_TEXT.

VI. Flowchart of Encryption Algorithm

The flowchart of encryption algorithm is:

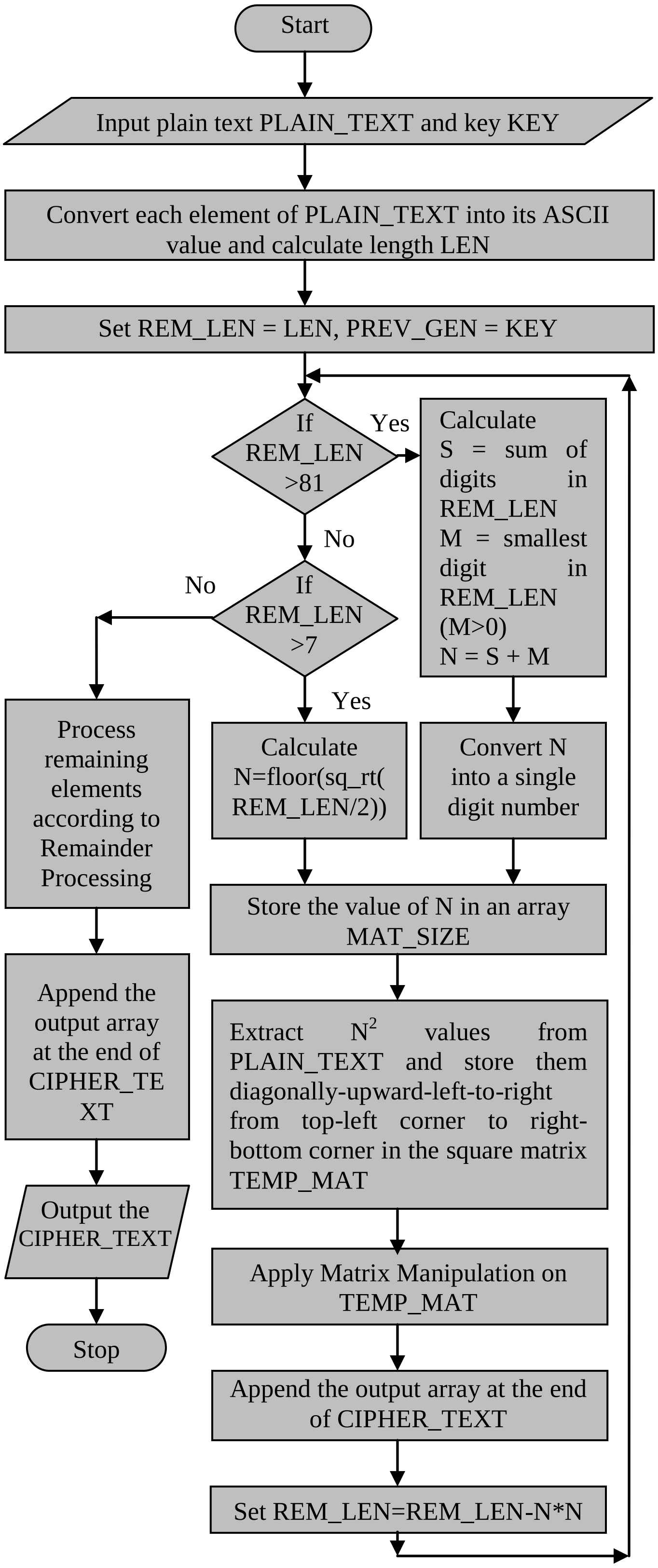


Fig. 1. Flowchart of Encryption Algorithm

The flowchart of Matrix Manipulation for encryption is:

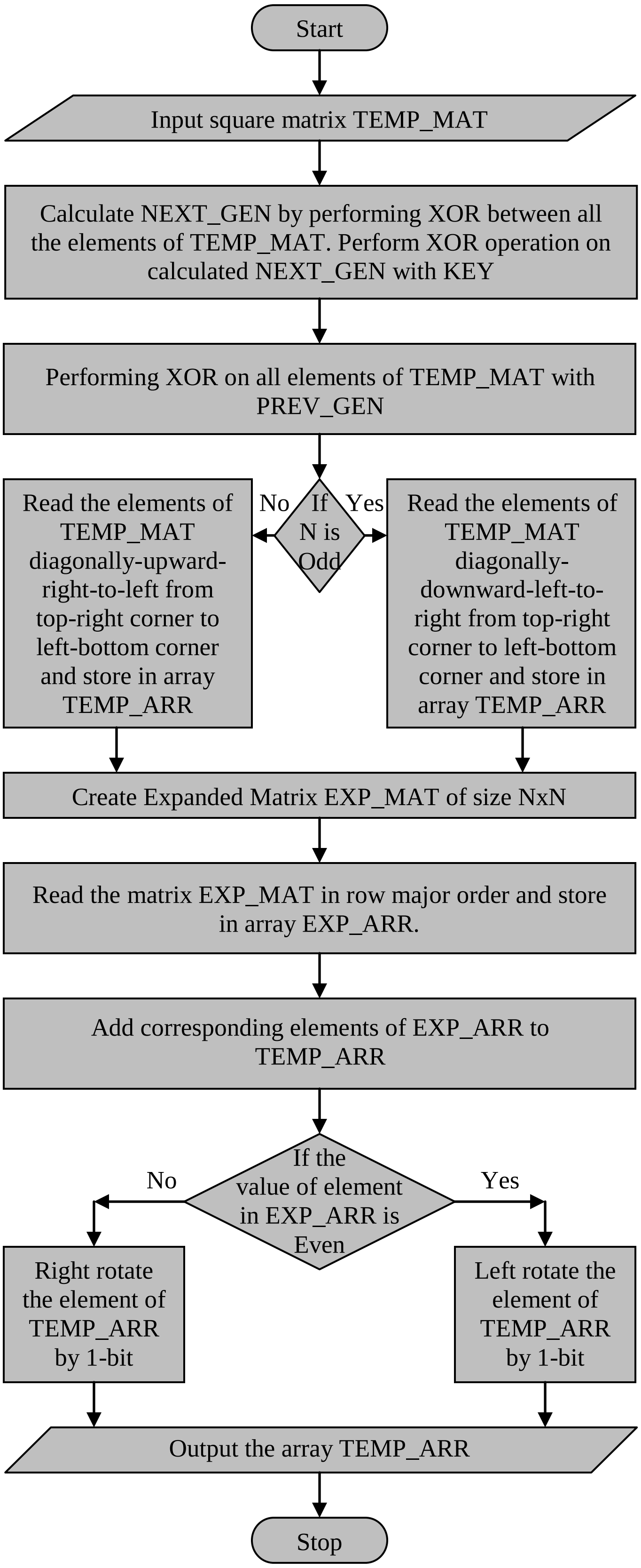


Fig. 2. Flowchart of Matrix Manipulation for Encryption

The flowchart of Remainder Processing for Encryption is:

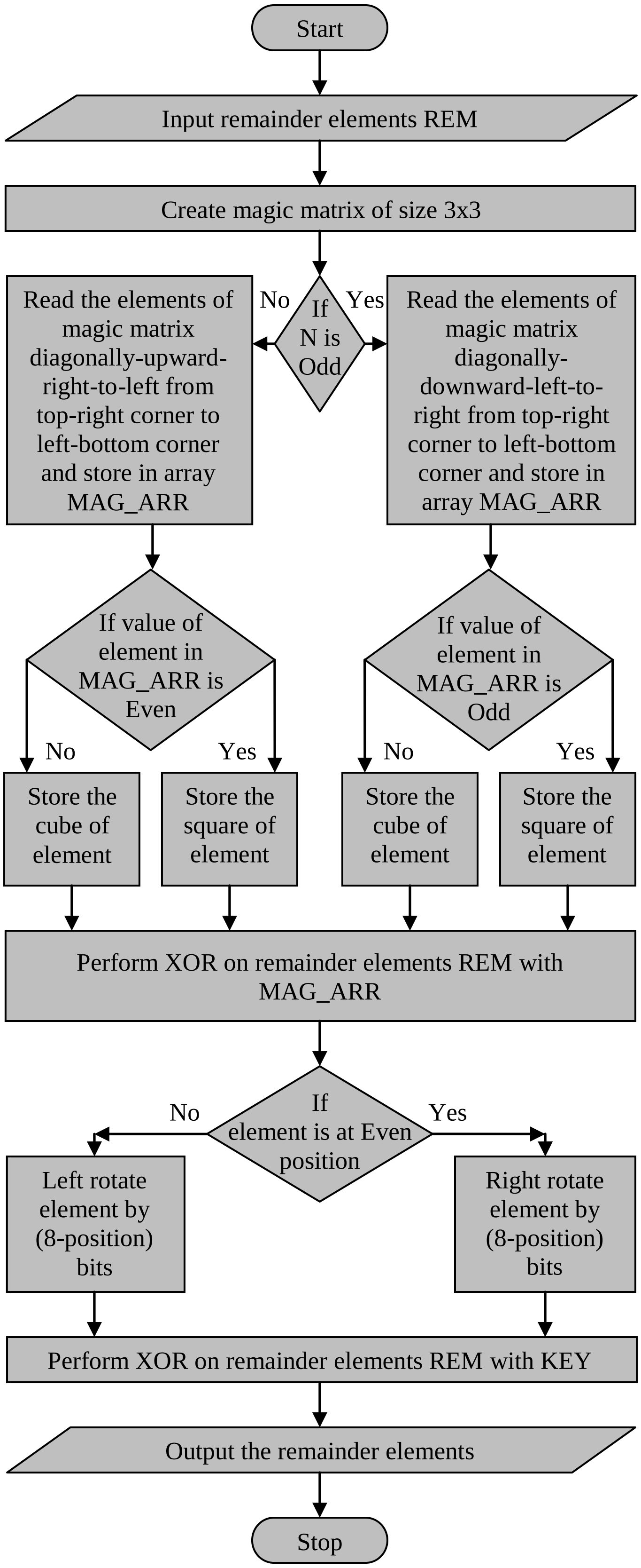


Fig. 3. Flowchart of Remainder Processing for Encryption

VII. FLOWCHART OF DECRYPTION ALGORITHM

The flowchart of decryption algorithm is:

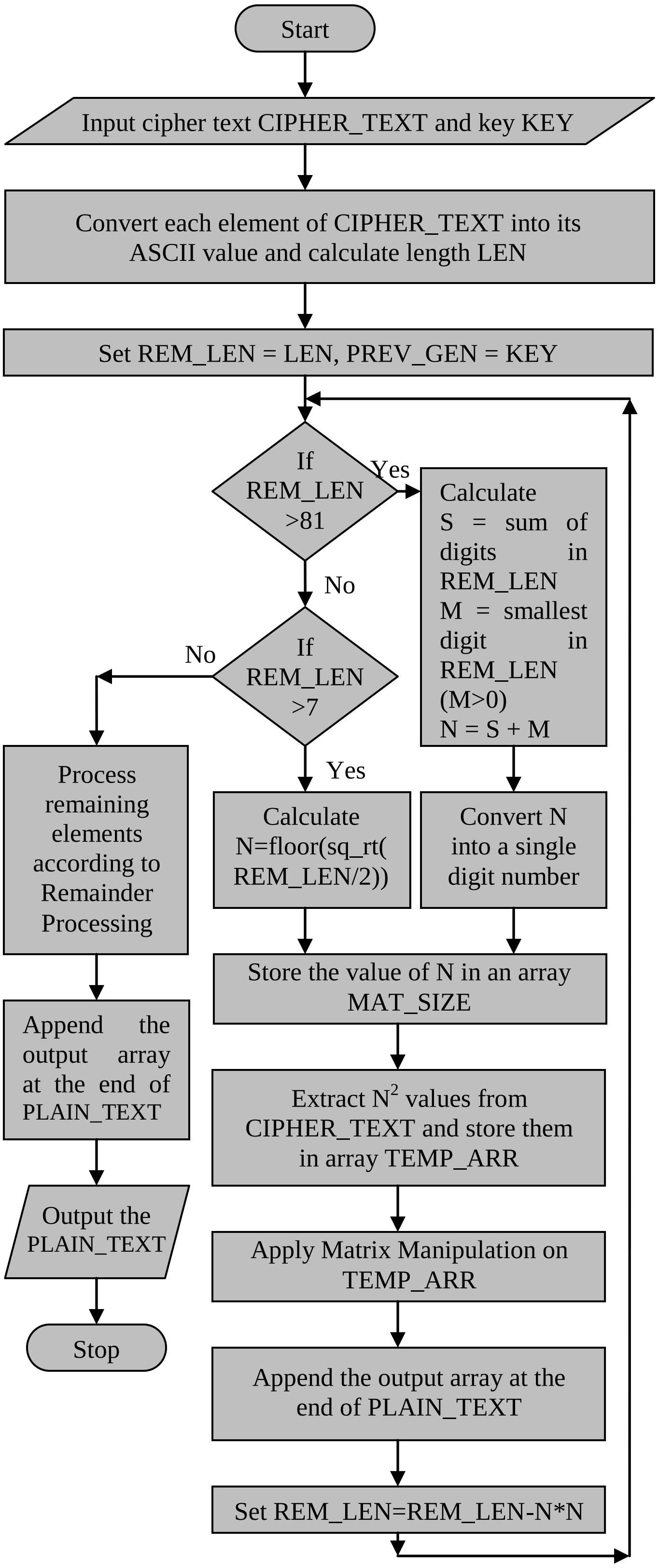


Fig. 4. Flowchart of Decryption Algorithm

The flowchart of Matrix Manipulation for Decryption is:

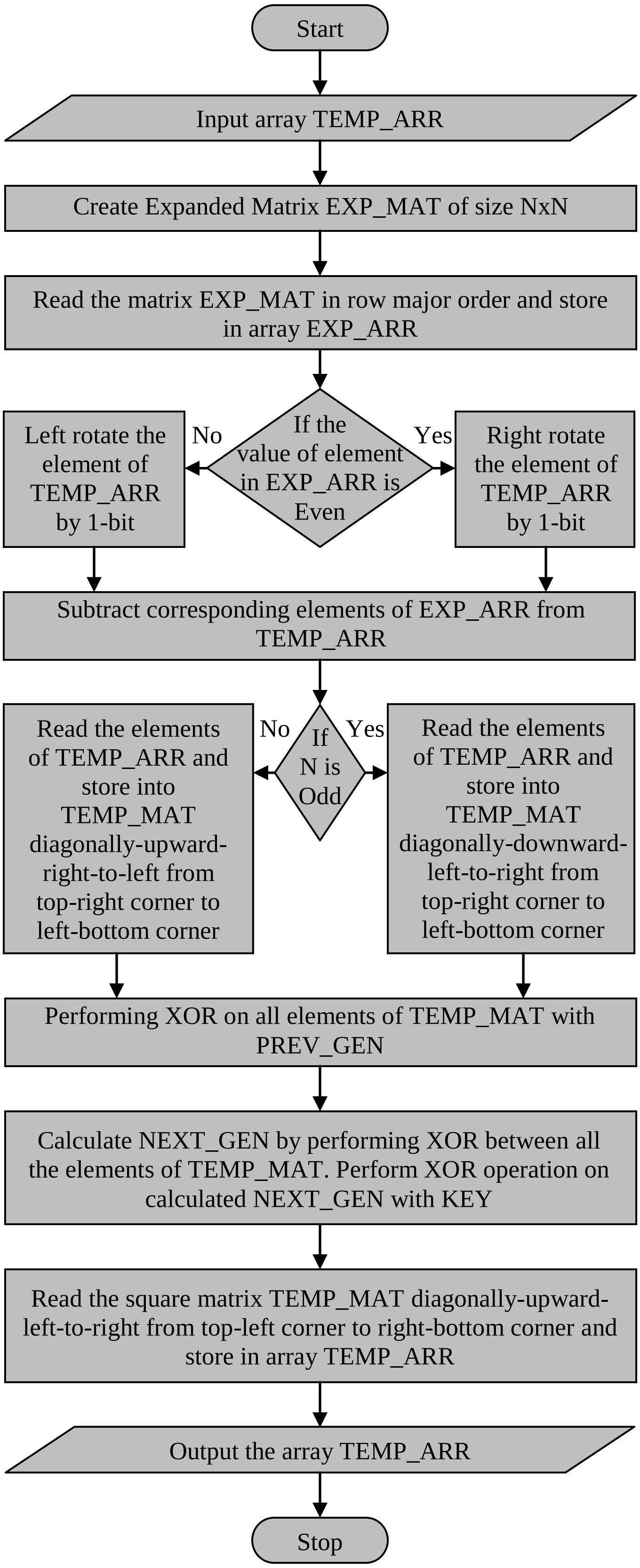


Fig. 5. Flowchart of Matrix Manipulation for Decryption

The flowchart of Remainder Processing for Decryption is :

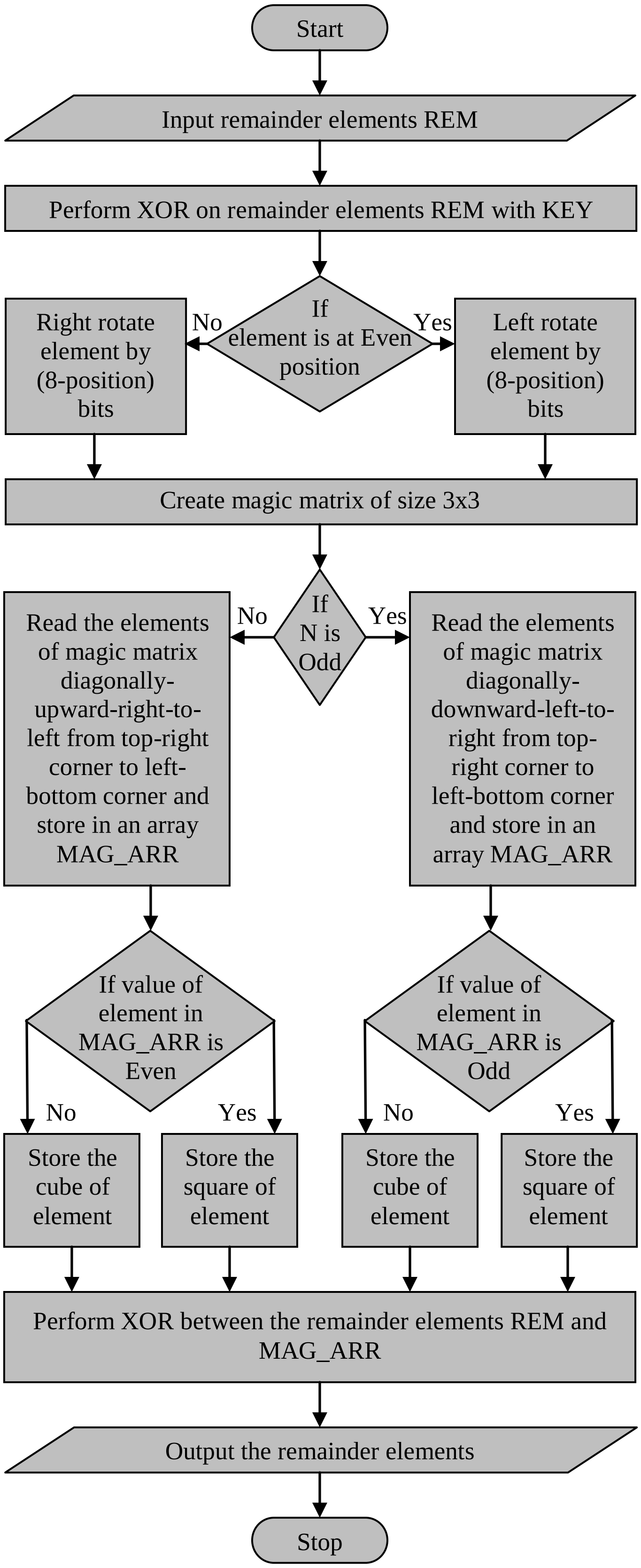


Fig. 6. Flowchart of Remainder Processing for Decryption

The flowchart of Expanded Matrix is:

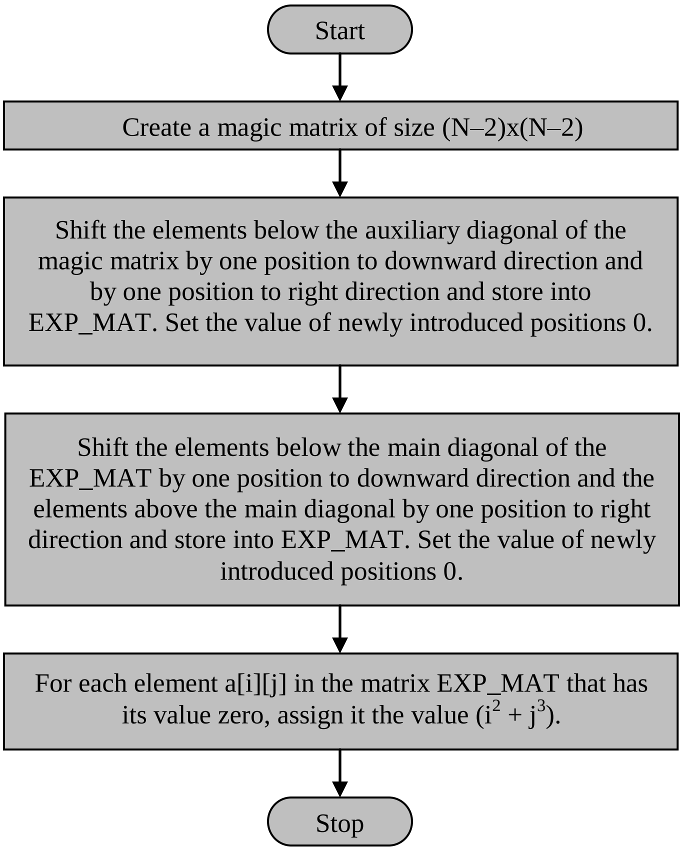


Fig. 7. Flowchart of Expanded Matrix

VIII. RESULT

On applying the proposed algorithm on different strings of varying length, the results obtained are astounding and noteworthy. The structure of statements in the plain text has been drastically changed. Some examples of the string length, key used, and encryption time are summarized in the table 1.

TABLE I. RESULTS OF SEMR ENCRYPTION ALGORITHM

Length of String	Key	Time (in seconds)
8	11	0.267490
109	77	0.583583
150	241	0.691474
200	227	0.832776
250	245	0.933613
300	149	1.061114

On changing the key for the same input string, the resultant cipher text is completely modified and cannot be correlated, but the encryption time does not change by a significant amount. This proves the dynamism of this algorithm. Some examples of the plain text “This is very secret message.” with different keys, encryption time and the resultant cipher text are summarized in the table 2.

TABLE II. RESULTS OF SEMR ENCRYPTION ALGORITHM ON VARYING KEY

Key	Time (in seconds)	Cipher Text
29	0.379354	ì¾²'Eþ¿%h`¨^¢7¼®%:¶ • +Ý
69	0.335716	\ â)n£´Ø¿3·Ä¾¦ªîÕIsÞ
119	0.378893	@æÆRP´¦Ö®´»²·p£Üç{A·ì
159	0.379310	ñÿ·h þ eÚÚeçcã2Áïf?4©_
209	0.332850	uàEûcÙ • õ • éç¿%ÐGÓzAÝçJ
249	0.348509	%ÌëYËW7Í]±é»íúðouÜSRiõÏ9b

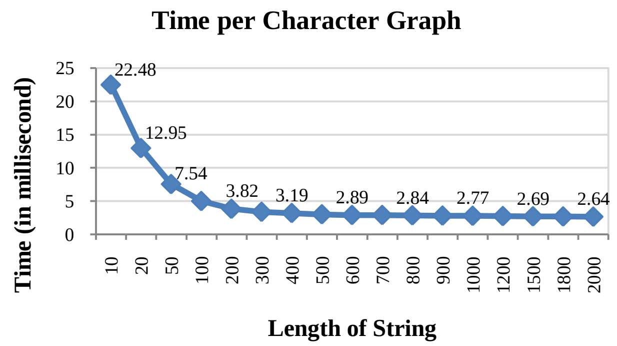


Fig. 8. Time per Character Graph

On plotting the time taken for encryption of each character of the string of a particular length, against the length of the string, we obtain the above Time per Character Graph.

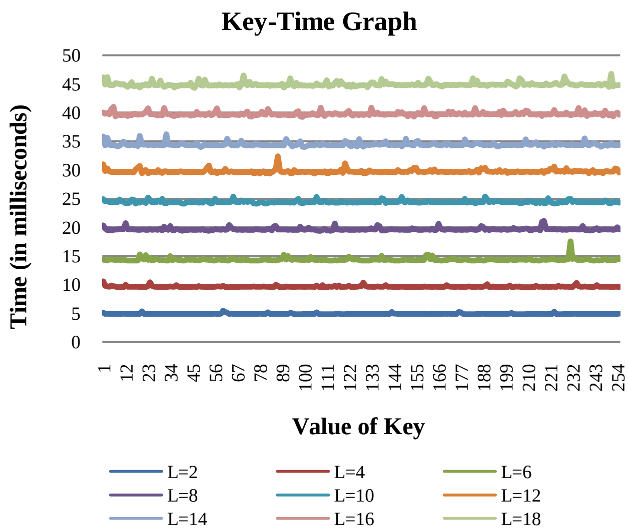


Fig. 9. Key-Time Graph

On plotting the time taken for encryption of the string, of small length, against different keys used for encryption, we obtain the above Key-Time Graph.

IX. Conclusion

In the proposed work, we have introduced a new technique of breaking the string into numerous parts before performing encryption. The strings which can directly break into the square matrix have been processed in such a way that they do not form the pit-holes in the algorithm or compromise with the security. The string, being broken into parts and having been separately encrypted, does not form a peculiar pattern that could be easy to recognize. The method of reading and placing the data into the matrices is different and changes rapidly, since it is not mere row-major order or column-major order. The frequency and value of characters is altered by using expanded matrix. The use of diagonal upward direction and diagonal downward direction is random and the creation of expanded matrix is unexpected and is impossible to be guessed even by hit and trial method. The key received from the user is used to hide the characters after performing some manipulation of the key. If wrong key is used, it would be impossible to break the cipher. On changing the key, the algorithm will dynamically change on itself. The encryption, and therefore, decryption of the successive matrices is linked, hence, until the present matrix is decoded perfectly, the next matrix cannot be decoded. The use of left and right rotation changes the data completely. Various diverse operations being executed on the string, changes the structure of the sentence.

The placement of the steps and operations is done in such a way that, mixing of all the steps is complicated and is therefore difficult to guess. Even if the different steps and operations are identified, placing them in correct order is very crucial and is therefore complex. The conditions applied at various steps allow the procedure to follow different set of steps for different strings.

X. Future Scope

This algorithm introduces a technique of breaking the string into small pieces before performing any operation, and is itself sufficient to provide the confidentiality at reasonable computation rates. The algorithm may be manipulated by changing several calculations, conditions and operations to make a stronger, more reliable and highly erratic algorithm.

References

[1] Vishwa Gupta, Gajendra Singh, and Ravindra Gupta, "A Hyper Modern Cryptography Algorithm to Improved Data Security: HMCA," International Journal of Computer Science & Communication Networks, vol 1(3), ISSN: 2249-5789, pp. 258-263.

[2] Vishwa Gupta, Gajendra Singh, and Ravindra Gupta, "Advance cryptography algorithm for improving data security," International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, vol. 2, issue 1, January 2012.

[3] Manas Paul, and Jyotsna Kumar Mandal, "A Universal Bit Level Block Encoding Technique Using Session Based Symmetric Key Cryptography to Enhance the Information Security," International Journal of Advanced Information Technology (IJAIT), vol. 2 no.2, pp. 29-40, April 2012.

[4] Manas Paul, and Jyotsna Kumar Mandal, "A General Session Based Bit Level Block Encoding Technique Using Symmetric Key Cryptography to Enhance the Security of Network Based Transmission," International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), vol. 2 no. 3, pp. 31-42, June 2012.

[5] Somdip Dey, Joyshree Nath, and Asoke Nath, "An Integrated Symmetric Key Cryptographic Method – Amalgamation of TTJSA Algorithm , Advanced Caesar Cipher Algorithm, Bit Rotation and Reversal Method: SJA Algorithm," I. J. Modern Education and Computer Science, DOI: 10.5815/ijmecs.2012.05.01, pp. 1-9, 2012.

[6] Niraj Kumar, Pankaj Gupta, Monika Sahu, and Dr. M A Rizvi, "Boolean Algebra based Effective and Efficient Asymmetric Key Cryptography Algorithm: BAC Algorithm," IEEE, 978-1-4673-5090-7/13, pp. 250-254, 2013.

[7] Gaurav Bhadra, Tanya Bala, Samik Banik, Asoke Nath, and Joyshree Nath, "Bit Level Encryption Standard (BLES) :Version-II," World Congress on Information and Communication Technologies, IEEE, 978-1-4673-4805-8/12, pp. 121-127, 2012.

[8] Akanksha Mathur, "A Research paper: An ASCII value based data encryption algorithm and its comparison with other symmetric data encryption algorithms," International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, vol. 4 no. 09, pp. 1650-1657, September 2012.

[9] Manas Paul, and Jyotsna Kumar Mandal, "A Novel Symmetric Key Cryptographic Technique at Bit Level Based on Spiral Matrix Concept," International Conference on Information Technology, Electronics and Communications (ICITEC – 2013), Bangalore, India, pp. 6-11, March 2013.

[10] Nehal Kandele, and Shrikant Tiwari, "A New Combined Symmetric Key Cryptography CRDDBT Using – Relative Displacement (RDC) and Dynamic Base Transformation (DBTC)," International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, vol. 2 Issue 10, pp. 1597-1603, October - 2013

[11] Subhranil Som, and Mandira Banerjee, "Cryptographic Technique by Square Matrix and Single Point Crossover on Binary Field," IEEE, 978-1-4673-2821-0/13, 2013.

[12] Suyash Kandele, and Veena Anand, "A Novel Cyclic-Lower-Upper-Rectangular (CLUR) Cryptography Method," International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, vol. 3 issue 11, pp. 329-335, November 2014.

[13] Ansar Ahemad Shaikh, and Nilesh S Vani, "An Extended Approach for Securing the Short Messaging Services of GSM using Multi-threading Elliptical Curve Cryptography," International Conference on Communication, Information & Computing Technology (ICCICT), Mumbai, India, January 2015.

[14] Gaurav Bansod, Nishchal Raval, and Narayan Pisharoty, "Implementation of a New Lightweight Encryption Design for Embedded Security," IEEE Transactions on Information Forensics and Security, vol. 10 no. 1, pp. 142-151, January 2015.

[15] Jongkil Kim, Willy Susilo, Man Ho Au, and Jennifer Seberry, "Adaptively Secure Identity-Based Broadcast Encryption With a Constant-Sized Ciphertext," IEEE Transactions on Information Forensics and Security, vol. 10 no. 3, pp. 679-693, March 2015.

[16] Dindayal Mahto, and Dilip Kumar Yadav, "Enhancing Security of One-Time Password using Elliptic Curve Cryptography with Biometrics for E-Commerce Applications," IEEE, 978-1-4799-4445-3/15, 2015.

[17] Subhas Barman, Debasis Samanta, and Samiran Chattopadhyay, "Revocable Key Generation From Irrevocable Biometric Data for Symmetric Cryptography," IEEE, 978-1-4799-4445-3/15, 2015.

[18] Harikrishnan T, and C. Babu, "Cryptanalysis of Hummingbird Algorithm with Improved Security and Throughput," International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SA TA), 978-1-4799-7926-4/15, 2015.

[19] Ali M Alshahrani1, and Stuart Walker, "Tesseract: A 4D symmetric key container for multimedia security," IEEE, ISBN: 978-1-4799-6376-8/15 pp. 139-142, 2015.

[20] Yoon Jib Kim, and Ki-Uk Kyung, "Secured Radio Communication Based on Fusion of Cryptography Algorithms," IEEE International Conference on Consumer Electronics (ICCE), pp. 388-389, 2015.

[21] Lidia Ogiela, and Marek R. Ogiela, "Insider Threats and Cryptographic Techniques in Secure Information Management," IEEE Systems Journal, 2015, in press.

[22] Divyanjali, Ankur, and Trishansh Bhardwaj, "Pseudo Random Bit Generation Using Arithematic Progression," Fifth International Conference on Advanced Computing & Communication Technologies, 2327-0659/15, DOI 10.1109/ACCT.2015.90, pp. 361-366, 2015.

[23] Prachi, Surbhi Dewan, and Pratibha, "Comparative Study of Security Protocols to Enhance Security over Internet," Fifth International Conference on Advanced Computing & Communication Technologies, IEEE, 2327-0659/15 DOI 10.1109/ACCT.2015.34, pp. 552-556, 2015