Journal of Computer Sciences and Applications, 2020, Vol. 8, No. 2, 56-61
Available online at http://pubs.sciepub.com/jcsa/8/2/3
Published by Science and Education Publishing
DOI:10.12691/jcsa-8-2-3

Dynamic Directional NxN Chart Based Text Based Substitution Cipher

Michael M Kangethe*, Elisha Odira Abade

School of Computing and Informatics, University of Nairobi, Nairobi, Kenya
*Corresponding author: mich01mk@gmail.com

Received August 11, 2020; Revised September 14, 2020; Accepted September 23, 2020

Abstract The need for simple secure substitution encryption algorithms has risen due to complexities derived from current encryption algorithms, which over-complicate the process through many sub-processes and consume more space than necessary. This also creates the need for a secure string encryption approach that can also be applied using a simple pen and paper to encrypt and decrypt messages between each other. In this paper we present a technique based on a Non-Deterministic directional *N*x*N* chart-based encryption cipher as a solution to this problem. This will be achieved by using a 2-dimensional *N* by *N* chart as the key while using a dynamically selected directional approach to generate both the cipher and plain-text.

Keywords: *encryption, decryption, ascii, strings, plain text, cipher text, multi dimension, hex, non-deterministic, random selection, text cipher, key chart, IV (Initialization Vector), XY (Horizontal/Row, Vertical/Column)*

Cite This Article: Michael M Kangethe, and Elisha Odira Abade, "Dynamic Directional NxN Chart Based Text Based Substitution Cipher." *Journal of Computer Sciences and Applications*, vol. 8, no. 2 (2020): 56-61. doi: 10.12691/jcsa-8-2-3.

1. Introduction

Cryptography is the practice and method for secure communication in the presence of third parties called adversaries. (Tanmoy and Ramkrishna, 2015)

Implementation of simple secure ciphers for text-based communication has become a necessity for efficiency of communication unlike other digital forms of communication. This has led to the development of several text-based ciphers based for this purpose.

The development of a randomized substitution cipher enhances the security of the cipher by obscuring the process of encryption and decryption of the data making it nearly impossible to decipher the data. This paper demonstrates how randomization can be used as a principle in substitution ciphers to enhance their security through the use of directional X, Y chart key-based character substitution.

2. Background Concept

Substitution is the replacement of one character with another. By using current text substitution ciphers the security of the data is usually weaker due to the limited number of possible representations of the ciphertext based on the plaintext. This forces cryptographers to build encryption algorithms that use multiple ciphers together with the substitution cipher primitive to strengthen the security of their developed encryption algorithms.

Randomization is a selection method base on pure chance alone and has been a reliable security concept when applied to cryptography as it introduces greater complexities to the process of encryption and decryption. The current known popular approaches of randomness apply in the generation of IVs. And recently in the use of encryption primitives.

Randomized text encryption algorithm proposed increases the complexity of cryptanalyst to decrypt the ciphertext and restricts them to break the security of encoded file. The proposed technique uses random numbers added to plaintext along with encryption key. After applying encryption technique, each time same plaintext will be converted to different ciphertext provided that encryption key is same or different [3].

Message Based Random Variable Length Key Encryption Algorithm. Dynamic and message dependent key generator was created by producing a random number and it was selected as the size of first chunk. Residual value of second chunk divided by first chunk concatenating with first chunk forms the first cipher as an input for SP-boxes. These processes repeated until whole message get involved into the last cipher. Encrypted messages are not equal under different run. Value of random number should be greater than 35 bits and plaintext must be at least 7 bits. A padding algorithm was used for small size messages or big random numbers [4].

Probabilistic Encryption aims to provide a practical implementation of a probabilistic cipher by extending on the algorithms by Fuchsbauer, Goldwasser and Micali. We provide details on designing and implementing the cipher and further support our understanding by providing a

statistical analysis of our implementation for the key generation, encryption, and decryption times taken by the cipher for key sizes of 1024, 2048, and 4096 bits for varying message spaces of 750, 1500, 3000, and 5000 bits. The concept of 'inter-bit operating time' is introduced for the cipher which calculates time elapsed between two instances of an operation [6].

3. Proposed Algorithm

The main approach to the solution is to create a substitution Cipher that uses a dynamically Generated Key Chart with randomly generated substitution values. Then use the Key Chart to navigate randomly in an X, Y direction to select the cipher substitution value for each plaintext character. The process should be reversible thus one should be able to Encrypt and decrypt the data given the Key Chart and the Directional sequence as the key.

In the encryption process only, the plaintext will be needed. The output of the encryption process will be three things namely:

1. Key Chart
2. Direction Sequence
3. Ciphertext

Both the Key Chart and the Direction Sequence can be combined to form a single key which will be used to reverse the encryption process and decrypt the ciphertext to attain the plaintext.

We assume the plaintext to be encrypted is in alphanumeric (***A-Z,0-9***) the first step will be to create a key chart that will contain the corresponding values of each character. This key chart will be used for the encryption and decryption process.

This is represented as mathematical process below:

3.1.1. Encryption Process

Given a 2 Dimension Chart *C* made of *N* by *N* hexadecimal character space we first get.

$$\boldsymbol{C}_{\{\boldsymbol{n},\boldsymbol{n}\}} = \{\boldsymbol{00}, \boldsymbol{01}, \ldots, \boldsymbol{FE}, \boldsymbol{FF}\}$$

Equation 1: Substitution Chart table Character space

And for each plaintext character p_i in the input text *P* the output E_i will become:

$$E_i = Z\begin{pmatrix} C_{xi} \\ C_{yi} \end{pmatrix}[p_i]$$

Equation 2: Substitution formular for each character in the Plaintext

Where *Z* is the Random Selection with respect to the substitution Chart's *X* and *Y* coordinates.

Using the chart as the substitution reference we follow the below rule to define the encryption process:

$$E(P|C) = SZ\left[\begin{pmatrix} C_X \\ C_Y \end{pmatrix}(P)\right].$$

Equation 3: The General Substitution formular for the Plaintext

Where:

E(P | C) is the Encryption of the plaintext *P* with respect to the substitution Chart *C.*

S is the substitution process

The output will be the ciphertext *E* and Substitution process stored as *KS.*

3.1.2. Decryption Process

Using the generated substitution mapping key from the process and stored as a key *Ks* we reverse the process by using the following process:

$$P_i = KS_I\begin{pmatrix} C_{xi} \\ C_{yi} \end{pmatrix}[e_i]$$

Equation 4: Substitution formular for each character in the Ciphertext

Where *ei* is the encrypted character at index *i* of the encrypted text *E.*

Thus, giving us the general formular

$$E(P|C) = SKS\left[\begin{pmatrix} C_X \\ C_Y \end{pmatrix}(P)\right]$$

Equation 5: The General Reverse Substitution formular for the Ciphertext

For the purpose of demonstration, the Algorithm will be described using a real-world example as detailed below

3.2. Example

Using the table below one should be able to encrypt my message 2341 For demonstration purpose we will use a small table of only four characters as our Key-space.

Table 1. Example Character Table chart

	1	2	3	4
1	A	B	C	D
2	B	C	D	A
3	C	D	A	B
4	D	A	B	C

The first row and column represent the plain-text mappings.

The data inside the table are the corresponding substituted representations of the plain-text data items at each instance.

3.2.1. Key-Chart Requirements

The column items have been placed in reverse unlike order to the rows so as to eliminate the possibility of cipher collision which is a case where multiple cipher approaches will lead to the same plain-text. This has also been applied at cipher-text key items $\mathbf{K_i}$ in the chart although the order is the same except at each row the item $\mathbf{K_i}$ appears at the $\mathbf{K_{i-1}}$ with full rotation for the outlying characters.

3.2.1.1. Note

- The first row will contain a random placement of all characters from in the alphanumeric set.
- The subsequent rows will contain the same set but with the characters shifted one position to the left or right.
- The overflowing keys will be placed in the starting points and this should follow until the key space cycle is complete.

3.2.2. Encryption

Assuming the system selected the second row at random and the directions selected was **H, V, H, V** where **H** represents the Horizontal direction and **V** represents the vertical directions. This can also be represented as **V=1** and **H=0**.

Therefore Using **2341** as our plain-text we encrypt it to become.

Steps:

- **2** will correspond to **C** [At column **2** and row **3**]
- **3** will correspond to **C** [At column **2** and row **3**] since we are selecting the corresponding row id Vertically,
- **4** will correspond to **A** [At column **4** and row **3**]
- And finally, 1 will correspond to **C** [At column **4** and row **1**]

So, the cipher-text **2341** will become **CCAC.**

3.2.3. Decryption

From the cipher-text above there is enough diffusion to eliminate its association to the plain-text

To decrypt the same text, we will need the chart and starting point which is H2 and the decryption process is represented as H, V, H, V

Therefore, Using **CCAC** as our cipher-text we Decrypt it to become.

Steps:

- **C** will correspond to the Column Key **2**[At column **2** and row **3**]
- **C** will correspond to the Row Key **3**[At column **2** and row **3**]
- **A** will correspond to the Column Key **4**[At column **4** and row **3**]
- and finally, **C** will correspond to the Row **1** Key **2**[At column 4 and row **1**]

Finally, the plain-text using the same hey and direction process of **CCAC** will become **2341.**

3.3. The Pseudo Algorithm

Variables
Random_Choices = "X","Y"
Direction_Selections[] =null
Cipher-text[] =null

3.3.1. Encryption Function

```
Function Encrypt():
{
Get Key_Chart,Current_x, Current_y,input_String
For i in input_String
        Direction_Selections[i]=GenerateRandomChoice
("X","Y")
        if(Direction_Selections[i] == "X")
        Cipher-text[i] =
Encrypt_Horizontal(Key_Chart,Current_x,Current_y,inpu
t_String[i])
Else
        Cipher-text[i] =
Encrypt_Vertical(Key_Chart,Current_x,Current_y,input_
String[i])
}
```

3.3.2. Decryption Function

```
Function Decrypt():
{
Get Key_Chart,Current_x,
Current_y,input_String,Direction_Selections[]
For i in input_String
        if(Direction_Selections[i] == "X")
        Cipher-text[i] =
Decrypt_Horizontal(Key_Chart,Current_x,Current_y,inpu
t_String[i])
        Else
        Cipher-text[i] =
Decrypt_Vertical(Key_Chart,Current_x,Current_y,input_
String[i])
}
```

3.3.3. X AND Y Encryption Functions

```
function
Encrypt_Horizontal(Key_Chart,Current_x,Current_y,Prev
_Y,Key)
{
For i in Key_Chart:
if(Key_Chart[0][i].equals(Key))
        {
                Current_x =i;
                Current_y=Prev_Y;
                Temp = Key_Chart[Current_y][i];
                break;
        }
Return Temp
}

function
Encrypt_Vertical(Key_Chart,Current_x,Current_y,Prev_Y
,Key)
{
For i in Key_Chart:
if(Key_Chart[i][0].equals(Key))
        {
                Current_x =X;
                Current_y=i;
                Temp = Key_Chart[i][Current_x];
                break;
        }
Return Temp
}
```

3.3.4. X AND Y Encryption Functions

```
function :
Decrypt_Vertical(Key_Chart,Current_x,Current_y,Prev_
Y,Key)
{
For i in Key_Chart:
if(Key_Chart[i][Current_x].equals(Key))
        {
                X_pos =Current_x;
                Y_pos=i;
                Temp =Key_Chart[i][0];
        }
Return Temp
}
```

```
function
Decrypt_Horizontal(Key_Chart,,Current_y,Prev_Y,Key)
{
For i in Key_Chart:
if(Key_Chart[Current_y][i].equals(Key))
        {
                X_pos =i;
                Y_pos=Current_y;
                Temp =Current_y[0][i];
        }
Return Temp
}
```

3.4. Security

To determine the algorithms security against brute force it was necessary to consider in the below factors.

3.4.1. Size of Data N

Key-space this is the Chart of Characters in the cipher in the PoC ranging from 0-9 and A-Z

$$s \in S \in A$$

Complexity is calculated using the below factors:

For a sentence **S** comprising of characters $s_0, s_1, s_2, \ldots\ldots, s_{n-1}, s_n$.

For every s_i in **S** complexity is **n** where **n** is the size of data **N** at each subset.

Thus, the complexity for the string is

Complexity $=S^2$.

3.5. Proof of Security

3.5.1. Crypt-analysis through brute-force of Possible Matches

Given 4 data items as the plain-text and cipher and 4 Possible Directions and 8 possible starting points we can use 0 to represent H and 1 to represent V and a binary table to list the possible outcomes as below.

Assuming we start with the first C in the first row as our default

Table 2. Possible Plaintext combinations from the Cipher (CCAC)

Possible combinations of the Cipher starting with C in the first row			
3	3	1	3
3	3	1	2
3	3	2	2
3	3	2	3
3	4	1	3
3	4	1	2
3	4	2	3
3	4	2	4
2	1	3	1
2	1	3	4
2	1	4	3
2	1	4	2
2	2	3	1
2	2	3	4
2	2	4	3
2	2	4	2
Possible Direction Combinations			
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Table 3. Possible Plaintext combinations from the Cipher (CCAC)

Possible combinations of the Cipher starting with C in the Second row			
2	2	4	2
2	2	4	1
2	2	1	4
2	2	1	3
2	3	4	2
2	3	4	1
2	3	1	4
2	3	1	3
3	2	4	2
3	2	4	1
3	2	1	4
3	2	1	3
3	3	4	2
3	3	4	1
3	3	1	4
3	3	1	3
Possible Direction Combinations			
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Table 4. Possible Plaintext combinations from the Cipher (CCAC)

Possible combinations of the Cipher starting with C in the Third row			
1	1	3	1
1	1	3	4
1	1	4	3
1	1	4	2
1	2	3	1
1	2	3	4
1	2	4	3
1	2	4	2
2	1	3	1
2	1	3	1
2	1	4	4
2	1	4	2
2	2	3	1
2	2	3	4
2	2	4	3
2	2	4	2
Possible Direction Combinations			
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Table 5. Possible Plaintext combinations from the Cipher (CCAC)

Possible combinations of the Cipher starting with C in the Fourth row			
4	1	2	4
4	1	2	3
4	1	3	2
4	1	3	1
4	4	2	4
4	4	2	3
4	4	3	2
4	4	3	1
1	4	2	4
1	4	2	3
1	4	3	2
1	4	3	1
1	1	2	4
1	1	2	3
1	1	3	2
1	1	3	1
Possible Direction Combinations			
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

From the above proof of tables, it is evident that for each combination of cipher-text there are N^2 possible plain-text options which increases the complexity of deciphering the text by itself. Where N is the size of the Cipher/Plain-text.

The correct plain-text from the cipher-text in the table also appears only once as highlighted in the second table. Thus, ensuring there won't be any collisions when deciphering the data.

4. Assumptions and Limitations of Scope

For the purpose of demonstration, the research and development of this algorithm focused solely on the decoded alphanumeric substitution process and not in any other extended digital data format and encoding or encryption primitives.

5. Conclusion

In this paper, we proposed an algorithm to encrypt the data based on random and dynamic Key Chat generation and substitution. The above system is produces reliable and stable results. However, the system can be improved by using Hex values for all types of data encryption. Keeping this in mind, the algorithm has been designed in a quite simple manner while focusing on high security.

References

[1] M. A. Murillo-Escobar, F. Abundiz-PÈrez, C. Cruz-Hern·ndez, R. M. LÛpez-GutiÈrrez. 2014. A novel symmetric text encryption algorithm based on logistic map.

[2] Mante, Pratik Gajanan, Oswal, Harsh Rajendra, Swain, Debabrata, Deshpande, Deepali. 2009. A Symmetrical Encryption Technique for Text Encryption Using Randomized Matrix Based Key Generation.

[3] Jamshed Memon, Mohd Zaidi Abd Rozan, Mueen Uddin, Adamu Abubakar, Haruna Chiroma, Dzurllkanian Daud. 2014. Randomized Text Encryption: a New Dimension in Cryptography. International Review on Computers and Software (I.RE.CO.S.), Vol. 9, N. 2

[4] Hamid Mirvaziri, Kasmiran Jumari Mahamod Ismail and Zurina Mohd Hanapi. 2009. Message Based Random Variable Length Key Encryption Algorithm. Journal of Computer Science 5 (8): 573-578, 2009

[5] Dhananjay S. Phatak, Qiang Tang, Alan T. Sherman, Warren D. Smith, Peter Ryan, † Kostas Kalpakis. 2014. DoubleMod and SingleMod: Simple Randomized Secret-Key Encryption with Bounded Homomorphicity

[6] Orhio Mark Creado, Yiling Wang, Xianping Wu, and Phu Dung Le. 2009. Probabilistic Encryption – A Practical Implementation, Fourth International Conference on Computer Sciences and Convergence Information Technology.