IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Evolutionary Relationship Of Some Substitution Ciphers

Dr. Antonio Silva-Santos

Assistant Professor

ISTEC-Lisbon Higher Institute of Advanced Technologies of Lisbon, Portugal

Abstract: Since the beginning of humanity, human beings have tried to communicate with each other, but whenever there is communication, there can be interference. So, to prevent interference from occurring in important communications, cryptography and steganography were created. Where encryption transforms the current text into non-perceivable text and steganography masks the information so that it is not perceptible. These techniques are thousands of years old, partly following different paths, with cryptography being, without a doubt, the one that has stood out the most over the centuries, as it is the one that has always been at the service of military leaders and rulers. Without a doubt, the most popular of all was the one called the Caesar Cipher, because it was used by Julius Caesar, Roman emperor around the 4th century BC. This, given its simplicity and versatility, gave rise to some others, such as: Vigenere, Beaufort, Gronsfeld, etc. Throughout this article we will study the evolutionary relationship of the Cesar cipher with those of: Vigenere, Beaufort and Gronsfeld**.**

Index Terms - ryptography, substitution cipher, monoalphabetic, polyalphabetic, Caesar Cipher, Vigenère Cipher, Beaufort Cipher and Gronsfeld Cipher.

I. INTRODUCTION

Cryptography has been used for thousands of years to help communicate confidential messages between mutually trusting parties [1]. Cryptography can be understood as the technique of transforming the original message (plain text) into an imperceptible form (ciphered text) for anyone who does not have the different variables and techniques to decode them. Sklavos and Zhang [2] write that cryptography is essentially the art of secret writing, while, Singh [3], writes that the purpose of encryption is not to hide the existence of a message, but to hide its meaning, a process known as encryption. Decryption is the opposite of encryption, that is, making the information visible again.

Currently, with the evolution of the means of communication, encryption is used in everything that requires security, hence the existence of safer and more efficient encryption/decryption methods, these methods are called ciphers.

Ciphers can be divided into two categories: symmetric and asymmetric, which, according to Aggarwal [4], depend on the type of security keys used to encrypt/decrypt data. Symmetric or single-key encryption uses the same key to encrypt and decrypt. In turn, symmetric encryption can be divided into two main categories, depending on its methodology: permutation and substitution [5]. According to Achary [6], the substitution cipher replaces one character in the plain text with another, while, Padhye et al. [8], writes that the transposition cipher the alphabets/characters of a given message, in plain text, are permuted to obtain the cipher text. In turn, plain text can be defined as the common text used by the sender and receiver of the message, on the other hand, cipher text is the message resulting from the transformation of plain text through an encryption algorithm. This document will focus on substitution ciphers: monoalphabetic and polyalphabetic. The monoalphabetic substitution cipher, according to Oppliger [9], each letter of the plaintext alphabet is replaced by another letter of the ciphertext alphabet, and the polyalphabetic substitution cipher means that multiple mappings of the alphabet are used, thus obscuring the relative frequencies of characters in the original plaintext [10].

II. CLASSICAL METHODS

As we go through the history of cryptography, we come across ciphers that evolve over time. In this article we will show the evolution in a set of known substitutions. First we will choose a description of them and in a second phase we will choose to show part of this evolution.

Caesar Cipher

The Caesar cipher is one of the oldest ciphers and a precursor to many other classical and modern ciphers. The Caesar cipher is named after the Roman dictator Julius Caesar (100-44 BC.), who, according to Suetonius, used it with a three-character advancement of the alphabet to protect messages of military importance. Although Caesar was the first to use this scheme, other substitution ciphers are known to have been used earlier. So messages were sent, replacing each A in the original message with a D, each B with an E, and so on through the alphabet [10].

The Caesar cipher can be easily modified so that the plaintext space and the ciphertext space are the set of all sequences $w = (w_1, w_2, \cdots, w_m)$, with $w_i \in \sum$, $1 \leq i \leq m$. In which, the key space is $\mathbb{z}_n$. The encryption function E_e replaces each letter w_i by $w_i + e \; mod \; n$ wi +e mod n, $1 \leq i \leq m$ [11].

Although Suetonius only mentions a Caesar shift of three places, it is clear that, using any shift between one and twenty-five places, it is possible to generate twenty-five distinct ciphers. According Singh [3], if one does not restrict oneself to changing the alphabet and allows the cipher alphabet to be any rearrangement of the simple alphabet, then one can generate an even greater number of distinct ciphers.

For any $k \in K$ and $x \in A$, where *K* is the set of characters that belong to $\mathbb{z}_n$ and *A* is the o alphabet used to encrypt/decrypt, the encoding and decoding process is defined as follows:

$$Encrypt: f\ (x) = \ (x + k) mod\ n. \qquad (1)$$

An encryption algorithm f replaces any letter with the one that occurs k positions ahead (cyclically) in the alphabet. f(x) is the ciphertext corresponding to the plaintext x.

$$Decrypt: f - 1(y) = \ (y\ - k)\ mod\ n. \qquad (2)$$

An f-1(y) encryption algorithm replaces any letter with the one that occurs k positions backward (cyclically) in the alphabet.

Taking the words of Achary [6], this cipher works using the operational model to encrypt and decrypt the messages, where the offset has k as the key, which is an integer from 0 to n. If the key is *K*, each letter in the plain text is replaced by the kth letter after the corresponding number (scroll right). Decryption for the given k performs the left scroll operation.

Vigenère Cipher

The Vigenère cipher, which according to Holden [12] was invented in 1553, is a polyalphabetic substitution cipher. This means that multiple alphabet mappings are used, thus obscuring the relative frequencies of characters in the original plain text [9]. This cipher is credited with a polyalphabetic system that uses standard alphabets and encrypts letter by letter using a short, repeated keyword; one of the simplest polyalphabetics to solve [13]. Vigenere's cipher was one of the best of its time, especially when using mixed alphabets; however, there are still several cases of breakage [14]. According to Stallings [15], this is the best-known polyalphabetic cipher and one of the simplest. Therefore, the set of related monoalphabetic substitution rules consists of the n Caesar ciphers with offsets from 0 to n. Each cipher is denoted by a key character, which is the ciphertext letter that replaces the plaintext letter a.

According to Stallings [15], taking as a simple attempt $P = p_0, p_1, p_2, \cdots, p_m$, and as a key $K = k_0, k_1, k_2, \cdots, k_m$. Then $c_0 = p_0 + k_{01} mod\ 26$, $c_1 = p_1 + k_1 mod\ 26$, $c_2 = p_2 + k_2 mod\ 26$, ⋯, $c_m = p_m + k_m mod\ 26$, with $C = c_0, c_1, c_2, \cdots, c_m$ the encrypted text being. To decrypt you use the reverse $p_0 = c_0 + k_0 mod\ 26$, $p_1 = p_1 + k_1 mod\ 26$, $p_2 = c_2 + k_2 mod\ 26$, ⋯, $p_m = c_m + k_m mod\ 26$.

Taking the previous paragraph as a basis, let's take: a sequence of simple text characters $X \subset A$, a string of encrypted text characters $Y \subset A$, and *K* a text character key. Then one can write the equations to implement the Vigenère encryption algorithm as follows:

$$Encrypt: f(x)\ = \ (x\ +\ k)\ mod\ n. \qquad (3)$$

In this formula, the function f transforms a character x, which represents each character of the plain text, with $x \in X \subset A$, depending on the respective character k of the key, with $k \in K$. To decrypt encrypted text:

$$Decrypt: f^{-1}(y)\ = \ (y\ -\ k)\ mod\ n. \qquad (4)$$

To decrypt the ciphertext, we use the inverse function above that transforms y, which represents each character of the encrypted text with $y \in Y \subset A$, depending on the corresponding character *k* of the key, with $k \in K$. The same key that is used to encrypt is the same key that is used to decrypt.

If we use the Vigenère table, the leftmost column of the table represents the key letters, while the top line represents the plain text letters, when we cross the line with the column we obtain the encrypted character [16]

Beaufort Cipher

The Beaufort cipher, known since around 1710, is a polyalphabetic substitution cipher and a variant of the Vigenère cipher, in which encryption and decryption are obtained using the same algorithm [17]. This is yet another periodic cipher, which is similar to Vigenere but using subtraction instead of addition [18]. For the plaintext letter *a*, the ciphertext letter *c* is the line containing the key *k* in column *a*. For the ciphertext letter *c*, the plaintext letter *a* is the column containing *k* in line *c* [19].

The Beaufort cipher can be described algebraically, using an encoding of characters as numbers and using the addition module *n*, where *a* is the characters that make up x_i the message and k_i is the key character, repeated if necessary. So the encoding and decoding of the message uses substitution:

$$f(x_i) = (k_i - x_i)\ mod\ n. \qquad (5)$$

Since the same function can be used to decrypt; that is, for the character y of the ciphertext, $f^{-1}(y_i) = (k_i - y_i)\ mod\ n$. The Beaufort cipher inverts the letters of the alphabet and then shifts them to the right by $(k_i + 1)$ positions, allowing f to be rewritten as follows: $f(a_i) = [(n-1) - a_i + (k_i + 1)]\ mod\ n$ (khan, 1973).

There is a variant of the Beaufort cipher that uses the substitution $f(a_i) = (a_i - k_i)\ mod\ n$. Since $(a_i - k_i)\ mod\ n = (a_i + (n - k_i))\ mod\ n$, the Variant Beaufort cipher is equivalent to a Vigenère cipher with key character $(n - k_i)$ [20].

This variant Beaufort cipher is also the inverse of the Vigenere cipher; so if one is used to encrypt, the other is used to decrypt.

Gronsfeld Cipher

The Gronsfeld cipher is a polyalphabetic substitution encryption method, created around 1744, that works like the Vigenère cipher, except it uses decimal number keys instead of letters. The key is made up of numbers, indicated by a quantity that is routed to letters of the alphabet where encryption is done from plain text to cipher text [21]. If the key is smaller in size than the text to be encrypted, it is repeated periodically with the intention that each plaintext character has a key value. Therefore, when the user enters a key that is shorter in length than the plain text, the key will automatically be repeated from the beginning until it reaches the length of the plain text [22].

Let *X* be a sequence of plain text characters $X \subset A$, *Y* an encrypted text string $Y \subset A$ and *K* a numeric key.

Taking into account what was previously described by the authors, the equations can be written to implement the Gronsfeld algorithm as follows:

$$Encrypt: f(x) = (x + k)\ mod\ n \qquad (6)$$

In this formula, the function *f* transforms a character x, which represents each character of the plain text with $x \in X \subset A$, depending on the number *k* of the key, with $k \in K$. To decrypt the Gronsfeld ciphertext:

$$Decrypt: f^{-1}(y) = (y - k)\ mod\ \text{n} \qquad (7)$$

To decrypt the ciphertext, we use the inverse function above that transforms y, which represents each character of the encrypted text with $y \in Y \subset A$, depending on the number *k* of the key, with $k \in K$. Make sure to use the same key for encryption and decryption, as per the theory of symmetric ciphers.

III. EVOLUTIONARY RELATIONSHIP OF CIPHERS

Although we have some precursor ciphers to the Caesar cipher, such as Hebrew, Greek and other pre-Roman civilizations. In this study, the Caesar cipher will be used as a basis, since the evolution is more visible than starting from others prior to the so-called Caesar cipher.

According to Jain [23] and according to Suetônio, Cesar used a shift cipher with a constant left shift of 3 characters to encrypt. To decrypt, the opposite is done, that is, a shift of 3 characters to the right, and whenever these shifts exceed the extremes, the characters were replaced by characters from the other extreme, taking the excess as a shift. We can describe this movement as cyclical, which was described by Alberti. According to Dooley [13], Alberti used two circles, one on top of the other, with the same alphabet in which they rotated around a central point, in which Alberti used the outer circle of the encrypted disk as a

simple alphabet and the inner one as an encrypted alphabet. When it rotated, the number of displacements had the letter from the top circle reflected in the bottom circle, where the character on the bottom disk represented the coding of the top disk.

Starting from Cesar's formula for an alphabet of size *n*; $f(x_i) = x_i + 3\ mod\ n$, for all x_i belonging to $X = \{x_0, x_1, x_2, \cdots, x_m\}$, with m the number of characters of the text to be encrypted, contained in *A*, where x_i is an element $A = \{A_1, A_2, A_3, \cdots, A_n\}$, so if we take A_1 as *0*, A_2 as *1*, and so on A_n, as *n-1*, the function $f(x_i) = x_i + 3\ mod\ n$ returns a value between *0* and *n -1* which can then transform into $A_1, A_2, A_3, \cdots, A_n$. In the case of the Portuguese alphabet, n=26.

Generalizing Cesar's formula to $f(x_i) = x_i + k\ mod\ n$, with *k* less than *n-1* and greater than *1* (Achary, 2021). This offset value is called the key, as previously stated.

Taking as $K = \{k_0, k_1, k_2, \ldots, k_m\}$, where $k_0 = k_1 = \cdots = k_m$ = k, the previous formula can now be represented in the form: $f(x_i) = x_i + k_i\ mod\ n$.

To decrypt, use the inverse function which will be $f^{-1}(y_i) = y_i - k_i\ mod\ n$. Since $-k_i\ mod\ n$ is equal to $n - k_i\ mod\ n$, then: $f^{-1}(y_i) = y_i + (n - k_i)\ mod\ n$, with k_i varying between *1* and *n-1*.

Keeping the function $f(x_i) = x_i + k_i\ mod\ n$, and replacing ki not with a fixed displacement value, but with a sequence K', defined as input key, in which $K' = \{k_0, k_1, k_2, \ldots, k_s\}$ and *s* less than or equal to *m*, where, if *s* less than *m* then $K = \{k_0, k_1, k_2, \cdots, k_s, k_0, k_1, \cdots\}$, inserting elements of the sequence of *K'* from left to right until the length of *K* is m. k_i is an integer between *0* and *n-1* just like x_i. Both representing the numerical value of a character in the base cryptography alphabet. When this type of encryption occurs, we are dealing with the Vigenère cipher method.

Analogously to the proposal for the Cesar cipher, the inverse function is used to decrypt: $f^{-1}(y_i) = y_i + (n - ki)\ mod\ n$.

Beaufort used the Vigenère table to create a cipher, which they gave their name, that mathematically can be written: $f(x_i) = (k_i - x_i)\ mod\ n$. Taking, as in the Vigenère cipher, a sequence of characters (text) *X* transformed into a number between *0* and *n-1* in the natural order of the alphabet *A*, and K' a key made up of characters from the alphabet and transformed into a numeric value.

Let $f(x_i) = (k_i - x_i)\ mod\ n$, if we take the complement of x_i, $n - x_i$, then the function to be taken will be $f(x_i) = k_i - x_i)\ mod\ n = (n - x_i) + k_i\ mod\ n$. The Beaufort cipher has the advantage over the Vigenère cipher because the function it encrypts also decrypts.

If the key K' is an integer, the function $f(x_i) = x_i + k_i\ mod\ n$, we are faced with the Gronsfeld cipher. If *K'*, the sequence inserted as the input key, has a length less than or equal to m, then we will fill the sequence by repeating the values of the first sequence from left to right until reaching length m, as in the Vigenère cipher.

If we look at the encryption of the Gronsfeld cipher, we can see that it is the Cesar evolution, in which the key does not have a unique numerical offset, but one for each character to be encrypted. It follows that we will use the same inverse function that we used earlier, in Cesar and Vigenere: $f^{-1}(y_i) = y_i + (n - k_i)\ mod\ n$.

The advantage of studying this evolution helps in programming methods, making it more versatile.

IV. PROGRAMMING THE EVOLUTION OF CIPHERS

To program the evolution of the ciphers as described previously, the Python programming language will be used, describing whenever necessary what the functions do. We will also use n=26 which corresponds to the number of uppercase characters in the Portuguese alphabet.

The main function is: encode(text, key), which encrypts and decrypts plain or encrypted text. This function takes the text and key and returns the encrypted/decrypted text as per the input text and key. At the code level, it can be represented by:

```
def encode(text, key):
    textc=''
    for i in range(len(text)):
        textc +=chr(((ord(text[i])-65)+(ord(key[i])-65))%26+65)
    return textc
```

The chr() command converts a numeric value into its corresponding ascii character, while ord() converts a character into a number, which when subtracted from *65* (character *A* in the ascii code) gives a value between *0* and *25*. Finally applying the calculation of modulo *26 (%26)*, which will give a number between *0* and *25*, added to *65* and calculating the chr of this, will result in a character of the alphabet.

Before using this function, you need other functions that help prepare the text and key.

As we are only using the capital letters of the Portuguese alphabet: *A, B, ..., Z*; you have to clean up spaces, symbols, numbers, etc.; and transform everything into uppercase, both for plain text and for character keys. And for this purpose, the function was created:

```
def format_str(text):
    newtxt = ''.join([i for i in text if not i.isdigit()])
    return newtxt.replace(' ', '').upper()
```

A no less important function that follows is the function that takes the keys and places them in the same size as the plain or encrypted text, in which the size is identified and transforming this by repetition into a sequence of characters the size of the text to be used in the encode() function. This function: complete(word, length), receives a word and a length and returns a word of length equal to the length variable.

```
def complete(word, length):
    wordk = word * int((length/len(word)))
    length -= len(wordk)
    if length:
        wordk += word[:length]
    return wordk
```

As previously mentioned, the encode() function is used to encrypt and decrypt, as we saw previously: $f^{-1}(y_i) = y_i - k_i \ mod \ n$ is equal to: $f^{-1}(y_i) = y_i + (n - k_i) \ mod \ n$, then a function was created to perform the task of calculating the $(n - k_i) \ mod \ n$, as well as $(n - x_i) \ mod \ n$ in the case of the Beaufort cipher. This function was given the name: inverse(word), it receives a sequence of characters and returns its complement:

```
def inverse(word):
    iword=''
    for i in range(len(word)):
        iword +=chr((26-(ord(word[i])-65))%26+65)
    return iword
```

The last function to be used is only in the case of the Gronsfeld cipher, which converts a sequence of numbers into an alphabet sequence in which each character corresponds to the displacement value of the item in question.

```
def chr_convert(key):
    chrkey=''
    for i in range(len(key)):
        chrkey +=chr(int(key[i])+65)
    return chrkey
```

Finally, you will need some commands, such as entering text and printing the results, to demonstrate that the functions work and are in line with what was previously developed. First, enter the text and for each cipher, enter its key and calls the necessary functions for encryption and decryption:

```
txt_cod = input('Input text: ')
new_text=format_str(txt_cod)

print('')
print('********** Caesar Cipher **********')
key = input('Input Cesar Key: ')
ckey=chr(int(key)+65)
new_key = complete(ckey, len(new_text))
encod=encode(new_text, new_key)
print('  Encrypted text: {0}'.format(encod))
inv_key = inverse(new_key)
decode=encode(encod, inv_key)
print('  Plaintext : {0}'.format(decode))
```

```
print('')
print('********** Vigenere Cipher **********')
key = input('Input Vigenere Key: ')
txt_key=format_str(key)
new_key = complete(txt_key, len(new_text))
encod=encode(new_text, new_key)
print('  Encrypted text: {0}'.format(encod))
inv_key = inverse(new_key)
decode=encode(encod, inv_key)
print('  Plaintext : {0}'.format(decode))

print('')
print('********** Beaufort Cipher **********')
key = input('Input Beaufort Key: ')
txt_key=format_str(key)
new_key = complete(txt_key, len(new_text))
inv_text=inverse(new_text)
encod=encode(inv_text, new_key)
print('  Encrypted text: {0}'.format(encod))
inv_text = inverse(encod)
decode=encode(inv_text, new_key)
print('  Plaintext : {0}'.format(decode))

print('')
print('********** Gronsfeld Cipher **********')
key = input('Input Gronsfeld  Key: ')
ckey=chr_convert(key)
txt_key=format_str(ckey)
new_key = complete(ckey, len(new_text))
encod=encode(new_text, new_key)
print('  Encrypted text: {0}'.format(encod))
inv_key = inverse(new_key)
decode=encode(encod, inv_key)
print('  Plaintext : {0}'.format(decode))
```

The results meet expectations, as can be seen, using the text: “Before it is too late”, with the key words: “9” (Caesar Cipher), “big kiss” (nas cifras de Vigenère e Beaufort) and “76543” (Gronsfeld Cipher):

```
Input text: Before it is too late

********** Caesar Cipher **********
Input Cesar Key: 9
  Encrypted text: KNOXANRCRBCXXUJCN
  Plaintext : BEFOREITISTOOLATE

********** Vigenere Cipher **********
Input Vigenere Key: big kiss
  Encrypted text: CMLYZWAUQYDWGDBBK
  Plaintext : BEFOREITISTOOLATE

********** Beaufort Cipher **********
Input Beaufort Key: big kiss
  Encrypted text: AEBWROKIAORUEHBPC
  Plaintext : BEFOREITISTOOLATE

********** Gronsfeld Cipher **********
Input Gronsfeld  Key: 76543
  Encrypted text: IKKSULOYMVAUTPDAK
  Plaintext : BEFOREITISTOOLATE
```

Figure 1 Result of the simulation of the different ciphers

As can be seen from the figure 1, the desired result was achieved, and how encryption and decryption was carried out confirmed this. The keywords and text were chosen at random without any criteria, so as not to bias our results.

There is a difference in the input text in encryption and the output in decryption because as mentioned previously all characters have been transformed into uppercase and spaces removed. In order to see the input text as the same as the output, some small changes would have to be made and instead of using module 26, module 256 would be used.

V. CONCLUSION

As nothing in this world appears by chance and there is always a new theory about something previously discovered, the evolution of ciphers also occurs to this day, when many ciphers are based on classical ciphers.

Studying the evolution of some of the classic ciphers shows readers how they evolved and how they can take advantage of this evolution for practical problems that occur in our daily lives. In the case of the Vigenère ciphers, a greater evolution is observed in relation to the Cesar cipher than in the Gonsfeld and Beaufort ciphers in relation to Vigenère.

The optimization of the code demonstrated the versatility of the ciphers, which consolidates the Caesar cipher as the basis in this evolutionary sequence of ciphers.

This type of study can be extended to other ciphers, which will also lead beginners to better understand the connection between ciphers, as well as their natural evolution.

REFERENCES

[1] Stinson, Douglas R. and Paterson, Maura B. (2018) Cryptography: Theory and Practice CRC Press. Boca Raton, Florida, USA.

[2] Sklavos, Nicolas and Zhang, Xinmiao (2007) Wireless Security and Cryptography, Specifications and Implementations, CRC Press, Taylor & Francis Group, Boca Raton, Florida, USA.

[3] Singh, Simon (2001), The code book: how to make it, break it, hack it, crack it. Delacorte Press. New York, USA.

[4] Aggarwal, Surabhi (2016). A Review on Enhancing Caesar Cipher, International Journal of Research Science & Management, 3(6).

[5] Mihailescu, M. I. and Nita, S. L. (2021). Cryptography and Cryptanalysis in MATLAB: Creating and Programming Advanced Algorithms. Apress Media, New York, USA.

[6] Achary, R. (2021). Cryptography and Network Security: An Introduction. Mercury Learning and Information. Dulles, Virginia, USA.

[7] Padhye, Sahadeo; Sahu, Rajeev A. and Saraswat, Vishal (2018) Introduction to Cryptography. CRC Press, Taylor & Francis Group. Boca Raton, Florida, USA.

[8] Oppliger, Rolf (2021). Cryptography 101: From Theory to Practice. Artech House. Norwood, Massachusetts, USA.

[9] Thorsteinson, Peter and Ganesh, G. Gnana Arun (2004) .NET Security and Cryptography. Prentice Hall Professional Technical Reference, Pearson Education, Inc. Upper Saddle River, New Jersey, USA.

[10] Saha, Arijit et al. (2013). Information Theory, Coding and Cryptography, Dorling Kindersley, Pearson. New Delhi, India.

[11] Buchmann, Johannes (2004). Introduction to Cryptography, Second Edition. Springer-Verlag. Darmstadt, Germany.

[12] Holden, Joshua (2017). The mathematics of secrets : cryptography from Caesar ciphers to digital encryption. Princeton University Press. Princeton, NJ, USA.

[13] Dooley, John F (2018) History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms. Springer. Galesburg, IL, USA.

[14] Bauer, Craig P. (2013). Secret Story: The Story of Cryptology. Discrete Mathematics and its Applications CRC Press, Taylor & Francis Group, Boca Raton, Florida, USA.

[15] Stallings, William (2017) Cryptography and Network Security: Principles and Practice, 7th Edition, Global Edition. Pearson Education Limited. London, UK.

[16] Putri, Ranti Eka and Siahaan, Andysah Putera Utama (2018) Three-Pass Protocol Scheme using Gronsfeld and Vigenere Ciphers, International Journal of Scientific Research in Science and Technology.4(10) pp. 223-228

[17] Naing, Htet Htet and Aye, Zin May (2020). Innovation Security of Beaufort Cipher by Stream Cipher Using Myanmar-Vigenere Table and Unicode Table Proceedings of 2020 the 10th International Workshop on Computer Science and Engineering WCSE 2020), Yangon (Rangoon), Myanmar (Burma), pp. 52-56.

[18] Alallayah, Khaled et al. (2010). Attack and Construction of Simulator for Some of Cipher Systems Using Neuro-Identifier. The International Arab Journal of Information Technology, 7(4), pp365-372

[19] Kahn, D (1973) The CodeBreakers. Macmillan: New York

[20] Denning, Dorothy E. (1982). Cryptography and Data Security. Addison-Wesley Publishing Company, Inc. Don Mills, Ontario, Canada.

[21] Ridho, Abdurrahman etal. (2020). Optimization of The Gronsfeld Cipher Key Using Okamoto-Uchiyama Public-Key Cryptosystem for Data Security.3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology, pp 123-126.

[22] Zuhri Ramadhan, Andysah Putera Utama Siahaan (2018) Protection of Important Data and Information using Gronsfeld Cipher, International Journal for Innovative Research in Multidisciplinary Field, 4(10), pp. 128-132.

[23] Jain, Atish et al. (2015) Enhancing the Security of Caesar Cipher Substitution Method using a Randomized Approach for more Secure Communication. International Journal of Computer Applications. 129 (13), pp 6-11.