Ngram Analysis on Classical Ciphers for Ease of understanding of Cryptography

José E. Jiménez Correa
Computer Science
Alfredo Cruz, PhD
Electrical and Computer Engineering and Computer Science
Polytechnic University of Puerto Rico

Abstract – *An Ngram is defined as a group of characters in a text. They are used to determine the validity of texts in the decryption results and show how they can be affected. Using classical ciphers, we show how changes in their encryption and decryption processes can affect de data being processed. By analyzing the results and creating Ngram Stamps we determine the validity of the text. This paper will provide examples of how the Ngram detection can help validate texts and show how cryptography works in securing and validating your data using the processes of encryption and decryption.*

Key Terms – *Cipher, Classical Ciphers, Cryptanalysis, Ngram*

INTRODUCTION

A cipher is an algorithm used to encrypt and decrypt data, it is one of the methods used to secure and encrypt data on the net. It does not only have to encrypt data it can also be used to maintain integrity of the data being transmitted. Any kind of interruption on the cipher or method of decryption can prove fatal for the message or data. To illustrate this, we will use the classical ciphers since they give very clear and concise examples of these properties. These ciphers are the ones that can be resolved in very short times and where at the time of their creation top choices for encryption and decryption of data and messages. These classical ciphers are known as symmetrical ciphers[4] since they use the same key for encryption and decryption. They can also be broken by brute force which will be the method that we will use for the Ngram analysis. Thanks to this they are perfect for demonstration purposes.

Thanks to the advancement of technology the knowledge of how data is secured, managed, and encrypted is almost non-existent. This is a factor for people to make mistakes on their security thinking that they are protected. The common users just listen to all the features of an encryption service and decide that the one that has more options is the best. This is done without taking in consideration the security or integrity of the data since they lack the knowledge on how the processes of encryption and decryption works.

We show how the reliance of not knowing the processes of encryption and decryption can compromise the security and integrity of our daily life. By using classical ciphers as an example, we see how a change in the keys or processes of the ciphers can affect the data and its results. By explaining how encryption and decryption works using as basis the classical ciphers and explaining how the ciphers work, and how the results can be analyzed to get an idea on how a change in the cipher process can change the results and validate the data.

Definition of Concepts

The following concepts are of note and must be taken into consideration for the understanding of the article.

- *Ngram*: It is a sequence of letters that represents a sequence of characters.
- *Alphabet*: It is the defined collection of symbols that make up a valid language.
- *Symbol*: it is a predefined character that represents a meaning.
- *Cipher*: it is a sequence of equations and methods used to encrypt or decrypt data.
- *Encryption*: It is the process of processing a valid text into a ciphered text.

- *Decryption*: It is the process of processing a ciphered text into a valid text.

Limits and Barriers

On this project we will be working with various limitations. This is due to the quantity of variations that can exist on the encryption and decryption ciphers. The first limitation is that we are going to use classical ciphers due to the ease of decryption, encryption, and analysis. A second limitation is one of language. The language that we will be using for all source text and analysis is the English language and even using the English language, we will be limiting the language to a subset of it. This subset is defined as all the letters of the alphabet only using the capitalizations of the these letters (A,B,C, …, X,Y,Z) this would mean a total of 26 recognized characters or letter which would be our symbols. These symbols would be the ones that will encompass our alphabet of use.

Another limitation would be that of the analysis, it would be based on the Ngrams for the English alphabet. The Ngrams used for the analysis [1] will be the ones that are repeated at least 100,000 times in English texts and all other iterations of less mentions are discarded. This is based in a study by Mayzner [2] which catalogs the positions and locations of different Ngrams in English texts. The Ngrams that will be used are the ones that are mentioned on any part of the word or letter sequence we refer to this as the wild card */* Ngrams.

The method that we will use is for the decryption is the brute force method since it shows the best way for understanding on what is happening with the cipher and its analysis. This will be done for all possible keys in the given cipher. Another limit is the keys that use words as a key they have been limited to the top 100 keys of most used words in the English alphabet.

Origin Analysis

To validate the results of the analysis for the various ciphers one must first get the Ngram stamp of the data. A Ngram stamp is defined as the result of the Ngram analysis on a given text, it is the composition of Ngram hits from 2Ngrams to 9Ngrams. The source text for analysis is a selection of text from the Wikipedia web page of the Rigel Star [3] this text is composed of 21,344 characters.

The text is composed of capital letters, small caps, numbers, and special characters. This Text is then saved on a text file(.txt) for ease of access exactly as it is copied from the source without any modification. For an analysis to be completed the source text must be modified in a way that our analysis data can recognized the texts. For this what we will do is take the source data text and modify it so that all the letters of the alphabet are in their capitalized forms. This is done without modifying any of the format on the text file. All other characters that are not part of our pre-defined alphabet are left as they are and not modified in anyway. They then are moved to the new modified source text in their predetermined formats as shown in Table 1.

Table 1
Original Text Versus Modified Text

Original
… Rigel as β Orionis (Latinized to Beta Orionis) was made by Johann Bayer in 1603. The "beta" designation is commonly given to the second-brightest star in each constellation, but Rigel is almost always brighter than α Orionis (Betelgeuse)…
Modified
… RIGEL AS Β ORIONIS (LATINIZED TO BETA ORIONIS) WAS MADE BY JOHANN BAYER IN 1603. THE "BETA" DESIGNATION IS COMMONLY GIVEN TO THE SECOND-BRIGHTEST STAR IN EACH CONSTELLATION, BUT RIGEL IS ALMOST ALWAYS BRIGHTER THAN Α ORIONIS (BETELGEUSE)…

As one can see the modification do not affect the format of the text just the letters. The Ngram stamp of the Origin/Source text is thus able to be retrieved as shown in Table 2.

Table 2
Origin Ngram Stamp

2Ngram	**3Ngram**	**4Ngram**	**5Ngram**
363	1569	2252	2013
6Ngram	**7Ngram**	**8Ngram**	**9Ngram**
1544	1099	715	427

This result is obtained by counting how many times the Ngrams are mentioned in the text at least once. Some examples of the Ngrams are shown in Table 3.

Table 3
Ngram Example List

2Ngram	**3Ngram**	**4Ngram**	**5Ngram**
UR, CA, EL, TA, LA	STA, CTI, ICA, IST, EAR	FORM, NING, ECTI, SOME, PORT	PLACE, ETWEE, BETWE, RIGHT, AGAIN
6Ngram	**7Ngram**	**8Ngram**	**9Ngram**
ENERAL, SYSTEM, RELATI, CTIONS, ECAUSE	ORMATIO, CERTAIN, INCREAS, RELATIO, SPECIAL	ALTHOUGH, RODUCTIO, ODUCTION, ASSOCIAT, MATERIAL	HARACTERI, ERNATIONA, NTERNATIO, RNATIONAL, INTERNATI

This Origin Ngram Stamp will give us a base for comparison on the other results of the cipher analysis. This will help us know which results are the correct answers in subsequent tests and analysis. This Ngram stamp results in a chart outlining the peaks of the mentions of the Ngrams in the origin text showing the expected max values and correct values of the Ngram stamp analysis and is shown in Figure 1. This analysis will continue until it has passed through all the Ngram lists.

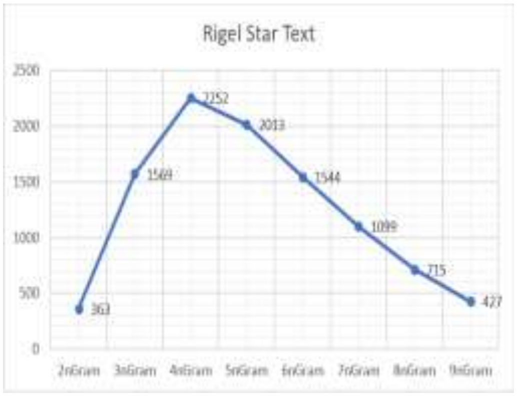


Figure 1
Origin Analysis

This Ngram stamp analysis consists of 669 Ngrams of two characters, 8653 Ngrams of three characters, 42171 Ngrams of four characters, 93713 Ngrams of five characters, 114565 Ngrams of six characters, 104610 Ngrams of seven characters, 82347 Ngrams of eight characters, and 59030 Ngrams of nine characters and are the ones that will be searched one by one on the subsequent analysis.

Caesar Analysis

To analyze the Caesar cipher, one must first understand it. It is a monoalphabetic cipher which means that all substitutions will be one on one. The Caesar cipher is defined as a shift in the alphabet of the letter to be encoded as shown in equation (1) or decoded using equation (2) by a predetermined amount being its key. One must note that to the Caesar cipher the letter does not matter, it is the position in the alphabet that does.

$$E_k(x) = (x + k) \bmod t \qquad (1)$$

$$D_k(x) = (x - k) \bmod t \qquad (2)$$

For the Caesar cipher one must select the alphabet that will be used, and we have selected the English alphabet and the subset of all the capitalized letters and their values which are shown in Table 4. In the equations the letter *t* represents the length of the alphabet which is *26*. The values range from the *0* to the *25* due to the design of the cipher algorithm.

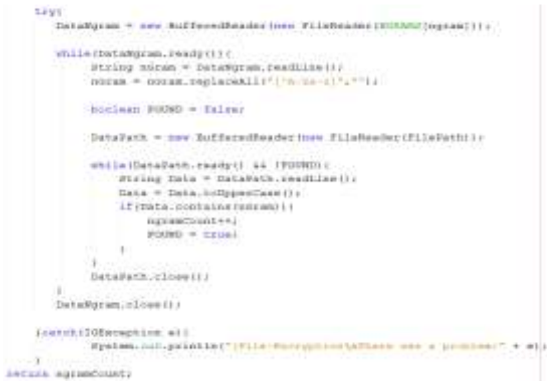
Table 4
Alphabet and its Values

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
00	01	02	03	04	05	06	07	08	09	10	11	12
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
13	14	15	16	17	18	19	20	21	22	23	24	25

In equations (1)(2) the *x* represents the value of the letter in the alphabet table, while the *k* represents the value of the key. During encryption, the key is added to the letter value according to the number represented in the alphabet. Once the equation is complete then the modulus operation is performed with the length of the alphabet to determine the correct value of the resulting equation. This result is the searched for in the alphabet shown in Table 4 to get the resulting value of the character. The process is the same for the decryption algorithm with the value of the letter minus the key. This value is then put through the modulus operation to get the correct letter in from the alphabet. If the result is a negative number, the length of the alphabet is added to simulate the wrap-around of the cipher.

To analyze the Caesar cipher, we must first generate the ciphered data of the Rigel Star Text which is the selected text for the Ngram analysis. We start by obtaining the original text and get a random key ranging from 0 to length of the alphabet. Then we capitalize the characters if possible and get the value of the letter and process to use the corresponding cipher on the text. If the result is a valid value proceed to encrypt the original text into cipher text following the encryption equation (1) if not the letter is not changed. We do that by trying to find if the selected character is part of our defined alphabet and if it is not then we return the selected character. The same is done with the decryption algorithm, if the resulting number is a negative number the length of the alphabet is added to the end to simulate the wrap around. After this process is done the resulting value which is the position of the character in the alphabet is searched for to provide the result. This will give us a semi-random ciphered text that will be used for analysis on the cipher.

We start the analysis by taking a ciphered text and brute forcing all the possible results based on the Caesar cipher which are based on the length of the alphabet, in the case of the English alphabet it is 26. This will give us a total of 26 results to analyze. The Caesar analysis algorithm works by using the properties of the cipher to test all possible values and analyzing all the results. By using the brute force method as the decryption method, the analysis will try all the possible keys for the ciphered text. Once a cipher text is completely decrypted by a given key it is then analyzed by the Ngram analysis algorithm.



```
try{
    DataNgram = new BufferedReader(new FileReader(NGRAMS[ngram]));

    while(DataNgram.ready()){
        String ngram = DataNgram.readLine();
        ngram = ngram.replaceAll("[^A-Za-z]","");

        boolean FOUND = false;

        DataPath = new BufferedReader(new FileReader(FilePath));

        while(DataPath.ready() && !FOUND){
            String Data = DataPath.readLine();
            Data = Data.toUpperCase();
            if(Data.contains(ngram)){
                ngramCount++;
                FOUND = true;
            }
        }
        DataPath.close();
    }
    DataNgram.close();

}catch(IOException e){
        System.out.println("[File Encryption]\nThere was a problem:" + e);
}
return ngramCount;
```

Figure 2
Ngram Analysis Algorithm

In Figure 2 we see the process of how the text is going to be analyzed. The analysis first starts by reading a Ngram from the Ngram lists and removes all non-alphabetic characters. It reads the Ngram and goes to find it in the text file being analyzed. It will search line by line until it has no more lines, or it finds a match for the Ngram. Once it finds a match it exits the search for that specific Ngram and add one to the Ngram counter. When all the Ngram list have been followed to the end it will create the Ngram stamp of the corresponding file adding all Ngram counters. All the results will then be joined together and analyzed to give us a more complete view of how the cipher behaved. This result will give us a sample of how this cipher algorithm works with erroneous keys and how the analysis determines the validity of the result in the Caesar cipher.

One of the notable things that must be taken into consideration is the time it takes to process all the analysis of the text; it took 3280.54 seconds and that is 54.67 minutes, almost an hour. This was for 26 keys that are part of the selected subset of the English alphabet only encompassing the capital letters. Each analysis took an average of 126.17 seconds per text and that is at least at a minimum 2 minutes per text to analyze. This analysis time is dependent on how the encryption works and the constraints implemented on the decryption, encryption, and analysis methods. The result of this analysis is demonstrated in the Figure 3.

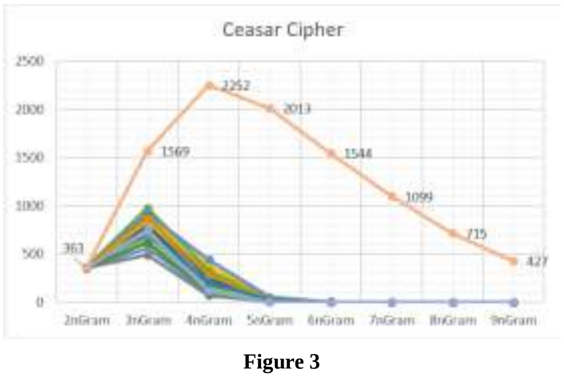


Figure 3
Caesar Analysis

In Figure 3 we can see how by using erroneous keys in the decryption mechanisms we can get erroneous results. This is an example of how encryption can secure your data. The Caesar cipher is a very weak encryption method currently since the calculation for the solution can be done in minutes if not hours and that is solely using brute force methods. By using specialized methods like the letter frequency, the solution can be found in minutes instead of hours while only producing the correct result without any error.

Affine Cipher

The affine cipher is a monoalphabetic cipher that uses two keys. These keys are determined by the length of the alphabet being used. To determine the keys, we use the length of the alphabet and multiply it by itself. This will give us the maximum possible number of keys according to the alphabet. To determine the valid number of keys we must first determine the valid values of the first key. For the first key to be valid it must be coprime with the length of the alphabet being used if not an error in encryption occurs.

In the encryption algorithm which is the equation (3) the letter *a* represents the first key in the algorithm to be used, the letter *b* represents the second key, the letter *t* represents the length of the alphabet, and the *x* the position of the letter to encrypt on the alphabet.

$$\mathrm{E}(\mathrm{x}) = (a*x + b) \bmod t \qquad (3)$$

$$\mathrm{D}(\mathrm{x}) = a^{-1}(x - b) \bmod t \qquad (4)$$

To encrypt a letter the first key is multiplied with the number value of the representation of the character to be encrypted and the result is then added with the second key. Then the result is passed through the modulus operation with the length of the alphabet t and the result looked for in the alphabet. To decrypt a character the algorithm used is the one determined in equation (4). In the equation the a^{-1} represents the multiplicative inverse respective to t which is the length of the alphabet. It is then multiplied to the result of the minus between the position of the character to decrypt and the second key. The modulus operator is then called to generate the corresponding position in the alphabet which gets the character to be returned.

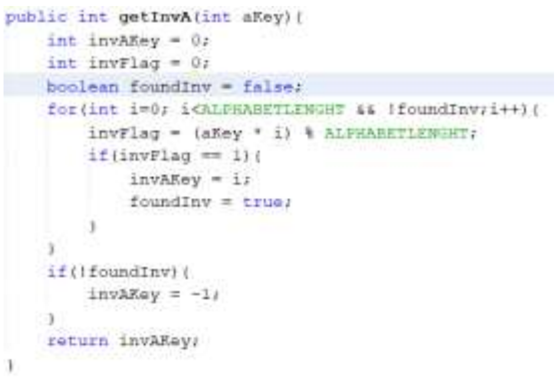
For the Affine cipher for it to be analyzed, we must first generate the ciphered data in which we will procced to analyze. For that purpose, we use the original text of the Rigel Star that has been selected and then encrypt it using a random valid set of keys. For the first key to be valid it must be coprime with the length of the alphabet which limits it to 12 which are: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.

The English alphabet has 676 different possible keys and of that only 312 are valid possibilities that we will be analyzing, the 12 possible values for the first key and the 26 possible values for the second key.

Once the keys and the text are acquired the process to encrypt the data begins. The data is verified to see if the characters are a valid entry and modified accordingly this means to capitalize all valid letters of the alphabet. The data is then encrypted using the affine cipher algorithm by multiplying the first key and the representation of the letter in numeric and then the result is added to the second key.

To process the decryption algorithm, the multiplicative inverse of a key selected must be found. The Figure 4 is an algorithm that allows us to find the inverse of a key according to the length of the alphabet using the length as a stopping point and a value to determine if it has been found. Only if during the loop the value selected is an inverse of the key sent the value is returned if not it is not a possible key. If this value is valid then it returns to the decryption algorithm to be processed. Once verified that the value is not negative the correct character is returned according to the alphabet selected. If an inverse is not found, then an invalid result is returned.

We start the analysis by taking the randomly generated ciphered text and starting the brute force decryption with all possible values for the English alphabet. That means using all 312 possible values. The algorithm works by using the already predetermined values of the English alphabet for the first key and the length of the alphabet for the second key alternatively. Once the data is decrypted it is then passed through the Ngram analysis algorithm so that it can give us the Ngram stamps.



```
public int getInvA(int aKey){
    int invAKey = 0;
    int invFlag = 0;
    boolean foundInv = false;
    for(int i=0; i<ALPHABETLENGHT && !foundInv;i++){
        invFlag = (aKey * i) % ALPHABETLENGHT;
        if(invFlag == 1){
            invAKey = i;
            foundInv = true;
        }
    }
    if(!foundInv){
        invAKey = -1;
    }
    return invAKey;
}
```

Figure 4
Inverse Algorithm Processing

Once the Ngram analysis is finished all the Ngram stamps will then be joined together and analyzed to give us a more complete view of how the cipher has behaved. This is demonstrated in Figure 5 which also gives us a sample of how the affine cipher algorithm works with erroneous keys.

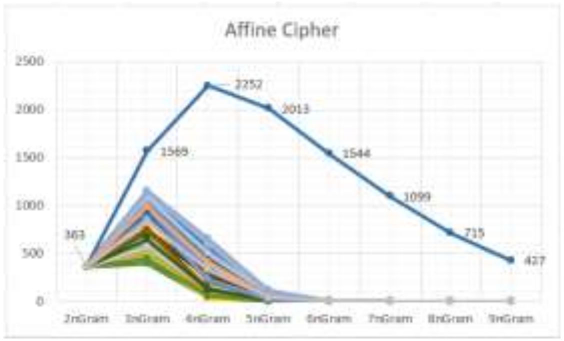


Figure 5
Affine Analysis

The time that it took for this cipher algorithm analysis to run is one of the things that must be considered due to the length of it. It took 37519.57 seconds and that is 625.33 minutes nearly 11 hours. This is due to the quantity of data and analysis that this cipher has and that is considering the length of the alphabet of 26 characters using only the capital letters. Each analysis took an average of 120.26 seconds per text and that is at least at a minimum 2 minutes per text to analyze. This analysis time is dependent on how the encryption works and the constraints implemented on the decryption, encryption, and analysis methods.

Here we can see how by using the erroneous keys in the decryption cipher an invalid result is generated. This example is like the Caesar cipher due to the similarities in their encryption and decryption algorithms. A Caesar cipher is an Affine cipher with a first key of a shift of one. The Affine cipher is a very weak encryption method since calculations at this age can occur very fast. By using the brute force method on the affine cipher one can see how many erroneous solutions can be generated. The waiting time can be reduced by using specialized tools like the letter frequency analysis determining the correct solution in a fraction of the time limiting the errors and minimizing the running time of analysis.

Vigenère Cipher

The Vigenère cipher is a polyalphabetic substitution cipher, it is based on joined Caesar ciphers with a different shift for every letter. The cipher encodes and decodes based on a given characters of text and a text key. This key will be repeated each time in a loop once its length is finished and the text continuous. This process can be defined in two different cipher algorithms. The encryption algorithm shown in equation (5) and the decryption algorithm shown in equation (6).

$$\mathrm{E(m)} = (M_i + K_{(i \bmod m)}) \bmod t \tag{5}$$

$$\mathrm{D(c)} = (C_i - K_{(i \bmod m)}) \bmod t \tag{6}$$

The M_i represents the character that is going to be encrypted in the position i and the K represents the key to be used in the position relative to the length of the key, while the C_i in the decryption algorithm represents the ciphered character in the text. This value of the letters is determined by the position of the letter in the alphabet. The m represents the length of the key and the t the length of the alphabet being used.

To use this cipher, we will start by defining the quantity of keys that we will be using since the combinations of keys are endless. Due to the key being based on a word or letter character sequence, we will limit it then to reasonable lengths. To select the keys to create a ciphered text for analysis we first select a limit for how many we will be comfortable for testing. The number we selected is 100 out of the 100000 that we have and that is the limit that we will test on this analysis using a randomized selection for the key. This process is done by selecting a random number between 1 and the limit key number. Then the algorithm opens the list of most mentioned words in the English alphabet and picks the selected word according to the key number randomly selected.

Once this key is selected from the random process it is sent to the cipher capitalized and modified to its numerical values. This is done so that when the key is being called it does not have to be converted to the format being used. These values will be saved in an internal key for ease of access due to the properties of the cipher.

In the process of encryption, it is the same as all the other ciphers. First one must first determine if the selected character is part of the alphabet by first capitalizing the selected character and looking if the character is part of its alphabet. If the character is not, return the same character to the stream, and if it is part of the alphabet then process the encryption algorithm with the given character by following the equation (5). Since we are using an internal key, we must call the function that will provide the correct key during encryption. The process of decryption will occur by using the equation (6). The process is the same just with a verification that if the result of the value is a negative number this result is shifted in the solution by adding the length of the alphabet.

We start the analysis by creating the ciphered data from the Rigel source text, it will be done using a random key from our selected list and limit. This ciphered text is the one that we will use for our analysis. Since the key is internal, we use a counter inside it to determine to correct key that will be returned. Thanks to this once an encryption, decryption, or analysis is completed that counter is reset to zero sot that any process starts with a reset counter.

We start by taking that ciphered text and deciding how many keys for the decryption we will be using. Our analysis consists of 100 keys due to the selected limitation. The brute force analysis of this cipher is done by processing the Vigenère cipher ciphered text with the given keys form the selected list of possible keys until the limit is reached or there are no more keys. Once a key is selected the counter for the key is reset and a new key is set.

Once a file is completely decrypted the Vigenère cipher Ngram analysis will start to create all the Ngram counters for the Ngram stamps to be used in the decryption results. These results will then be merged to show how the cipher algorithm has behaved with different keys and their tendencies.

This will give us the Ngram stamps to show us how the cipher works when using similar words, distinct words, and erroneous words as show in Figure 6. This analysis took 13434.70 seconds and that is 223.91 minutes, almost four hours. And this was for 100 predetermined keys part of the English alphabet using capital letters. It took an average of 134.35 seconds per key and that is around 2 and a half minutes. The analysis time is dependent on the length of the alphabet and the key.

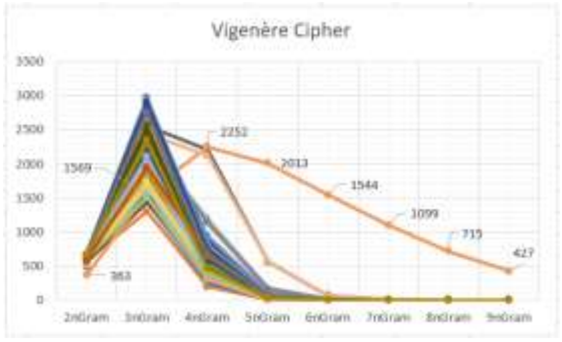


Figure 6
Vigenère Analysis

Due to the similarities of the key used in the decryption part of the analysis the Ngram stamp had some similarities in the beginning but we can see how more targeted Ngrams find the correct text. In this one we can see how the keys THAT, and THEN similar words to the key THAN can create errors and help approximate to the correct key. This is an example how today in this day it is recommended to create passwords completely different from any previous one to limit the paths of attack. One of the specialized methods to break this cipher is to use the Index of Coincidence and the Chi-squared statistic which will give us the possible length of the key and its distribution. This way the solution can be found in minutes instead of hours.

Playfair Cipher

The Playfair cipher is different from all other previous mentioned ciphers. This cipher works by using pairs of letters known as a bigram as an encryption and decryption method. For this cipher to work we use a modified Polybius square to encrypt and decrypt the text. This square is filled by using the text key and the alphabet of the language of the key. One example of this square is on Table 5. This is done by eliminating all repeated letters in the selected key and adding the missing letters form the alphabet the letter J and I interchangeable and one is omitted.

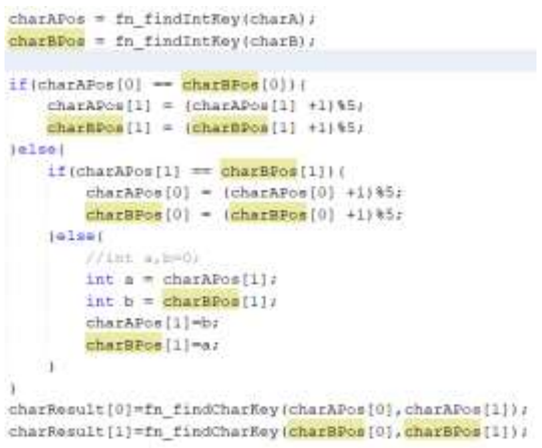
Table 5
Playfair Table Key Example

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

The key being used is “*PLAYFAIR*” with the English alphabet and the letter J is omitted since it is interchangeable with the letter I. All letters are used in their capitalized versions. The rules of the Playfair cipher [7] are:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter.
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively, if there are no letters the wrap around the row.
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively, of there a no letters then wrap around the column.
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. This is done following the order of presence.

These rules are the ones followed if you are encrypting a text. For the decryption process it is following the last three rules with the inverse instructions. We find the rule to follow by using this algorithm shown in Figure 7. This figure shows us the encryption algorithm of the Playfair Cipher. The first thing to do when starting this algorithm is to find the position of the given bigram in the Playfair key square table to encrypt. In this figure we see the main functions of the Playfair cipher.



```
charAPos = fn_findIntKey(charA);
charBPos = fn_findIntKey(charB);

if(charAPos[0] == charBPos[0]){
    charAPos[1] = (charAPos[1] +1)%5;
    charBPos[1] = (charBPos[1] +1)%5;
}else{
    if(charAPos[1] == charBPos[1]){
        charAPos[0] = (charAPos[0] +1)%5;
        charBPos[0] = (charBPos[0] +1)%5;
    }else{
        //int a,b=0;
        int a = charAPos[1];
        int b = charBPos[1];
        charAPos[1]=b;
        charBPos[1]=a;
    }
}
charResult[0]=fn_findCharKey(charAPos[0],charAPos[1]);
charResult[1]=fn_findCharKey(charBPos[0],charBPos[1]);
```

Figure 7
Playfair Rule Encryption Algorithm

The first rule is followed outside of the algorithm since it is only needed for the encryption part of the cipher. Thanks to these rules the Playfair cipher algorithm creates some modifications for the source text and its results. This first rule in the algorithm is managed so that all the letters identified as a ‘J’ are changed to a ‘I’ and only authorized letters are allowed in the alphabet. This algorithm also determines if the letter is a part of the alphabet or not and acts accordingly discarding all other non-valid characters not previously defined in the alphabet. Once this process is complete the process to encrypt or decrypt is then processed by following the algorithm. The cipher works by verifying the position in the array as rows and columns. The rows are determined in the first position of the array the ‘0’ and the column the ‘1’. The following rules are managed using their arithmetic properties. Once the rules are processed, they are sent to a function to determine their correct positions in the Playfair key table and return the correct characters.

The difference between the encryption and decryption algorithms is that the last three rules are followed in the reverse direction instead of adding one to the positions they are taken one and the exchange is done in reverse. During the decryption process if the resulting value is a negative number, then by following the properties of the cipher we can add 5 to the result to simulate a wrap-around of the cipher.

We start the analysis by selecting a random key form the top list of mentioned words for the English alphabet limiting it to the top 100 since is the limit that was selected. After we create the Playfair Key table we start encrypting the source text to a ciphered text. This text is modified in a way that eliminates all non-valid characters, this means all non-alphabetic characters are eliminated and only the new line character according to the algorithm is being noticed. This eliminates all spaces and special characters are present on the original text and will affect the analysis due to the change of the structure.

Once the cipher text is created, we start the brute force analysis of the cipher. We start by creating a Playfair table for all the keys to be used in the analysis one by one. Like the previous cipher, the keys would be read from a predetermined list until the limit. The Playfair brute force Ngram analysis starts by reading the results of the decryption process and performing the Ngram count for the Ngram stamp. Once the analysis is finished and the Ngram stamps have been collected we can see how the cipher analysis has been behaved. In Figure 8 we can see how all the Ngram stamps have behaved according to the results and limitations of the cipher.

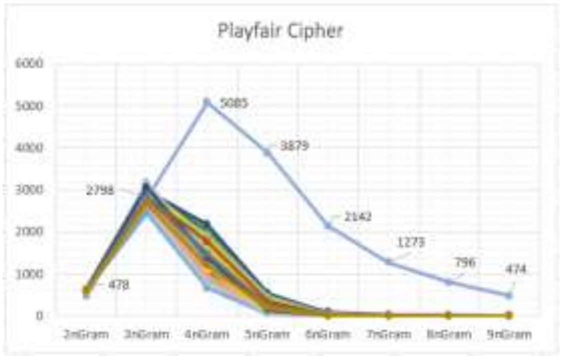


Figure 8
Playfair Analysis

The time it took for the analysis to finish is 8850.67 seconds and that is 147.51 minutes to process the 100 predetermined keys for the cipher. That is 2.26 hours of analysis time on an average pc. It had an average of 88.51 seconds for analysis, less than the previous due to the purging of non-valid characters during the encryption. We can see in the graphic that the mentions of Ngrams have risen in all the categories. This is due to the purging of the non-alphabetic characters in the text thus creating more false positives in the Ngram stamp. Thanks to the purging of non-valid characters the individual analysis has taken less time and the result becomes obvious due to the high quantity of results. One of the specialized methods to break this encryption is a frequency attack using bigrams.

CONCLUSION

By watching the result of the cipher algorithm and their analysis we can determine how secure they are and how they can ensure it. The ciphers show how when and erroneous method is used for a decryption of the text it gives us garbage data. This can tell us how secure the data is and how its integrity and availability is. At this age, these classical ciphers are not very secure with the advent of the computer era due to the processing power of the machines, but it gives us an idea of how the process works. We can see how the source text is processed into cipher text with the given ciphers and how changes in the method can give us different results.

Depending on the cipher a change of letter or change of number greatly affects the consistency of the message. By not following the correct steps of encryption or decryption the data that is to be processed changes greatly in response to wrong input. By using an incorrect key in the decryption of the analysis the analysis shows a garbled text and becomes and erroneous solution.

The analysis shows that by having a high count of valid Ngrams in the Ngram Stamp we can determine if the text is a valid solution or not. We can see how the analysis show a high count for nearly all the beginning Ngrams and becomes less as the Ngram length increases. In the Table 6 we can see how the average erroneous Ngram count works versus the total Ngram counts and how the correct Ngram stamp compares. On the source line we see the total of Ngrams in the list to find in each text and below the average of the Ngrams found on the erroneous texts.

Table 6
Total Ngrams Vs. Average Vs. Source

Source:	**2nGram**	**3nGram**	**4nGram**	**5nGram**
Rigel Text	***363***	***1569***	***2252***	***2013***
Total	*669*	*8653*	*42171*	*93713*
Caesar	359.32	762.60	201.68	15.40
Affine	359.24	785.68	234.23	22.54
Vigenère	625.01	2131.70	559.99	55.15
Playfair	610.32	2851.95	1318.16	190.12
Source:	**6nGram**	**7nGram**	**8nGram**	**9nGram**
Rigel Text	***1544***	***1099***	***715***	***427***
Total	*114565*	*104610*	*82347*	*59030*
Caesar	0.44	0.00	0.00	0.00
Affine	0.96	0.03	0.00	0.00
Vigenère	3.29	0.18	0.01	0.00
Playfair	15.17	1.40	0.26	0.04

Seeing this we can determine that the first two and last two Ngrams can be eliminated from the analysis since they either provide many false positives or does not provide enough positives. With this we can shorted the time of Ngram Analysis showing where a text can be valid or not. This can be understood since the average length of a word is from 4 characters to 5 characters. We can also determine the consistency of the data [6] by noticing that if a modification is used on the cipher, then the corresponding message will be erroneous. With the classical cipher we can see how an encryption and decryption works and we can have an idea of how difficult it is to break the encryption if you do not have a correct starting point. This can show us were the security is weak and can be broken and show how a better understanding of cryptography methods can protect our data in this age of information technology [5]. By using high level encryption, we can secure the data, but one must always take not that as technology advances the data can become less secure and more methods must be found to secure the data.

References

[1] Norvig, P. (2012). *English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU.* Retrieved September 24, 2019, from https://norvig.com/mayzner.html

[2] Mayzner, M. S., & Tresselt, M. E. (1965). *Tables of single-letter and diagram frequency counts for various word-length and letter-position combinations [by] M.S. Mayzner and M.E. Tresselt*. Goleta, Calif.] Psychonomic Press.

[3] Rigel. (2020, November 01). Retrieved January 1, 2021, from https://en.wikipedia.org/wiki/Rigel

[4] Quade, P. (2019). *The digital big bang: The hard stuff, the soft stuff, and the future of cybersecurity*. Indianapolis, IN: Wiley.

[5] Pfleeger, C. P., & Pfleeger, S. L. (2012). *Analyzing computer security: A threat/vulnerability/countermeasure approach.* Upper Saddle River, NJ: Prentice Hall.

[6] Whitman, M. E., &amp; Mattord, H. J. (2005). *Principles of information security 2nd. ed.* Boston, MA: Thomson Course Technology.

[7] Stallings, W. (2011). *Cryptography and network security: Principles and practice*. Boston: Prentice Hall.