CRYPTOBOX
User’s manual

Version 1.3

Lizeth Ortiz, Diego Romero,
Sofía Salinas, Oscar Rodriguez

Contents

1 Introduction 2

2 Instructions for use 3
- 2.1 Home Page 3
- 2.2 Shift Cipher 3
- 2.3 Permutation Cipher 4
- 2.4 Vigenère Cipher 5
- 2.5 Affine Cipher 6
- 2.6 Substitution Cipher 7
- 2.7 Hill Cipher 8
- 2.8 Simplified-DES Cipher 8
- 2.9 Triple Des Cipher 9
- 2.10 AES Cipher 9
- 2.11 RSA Cipher 10
- 2.12 Rabin Cipher 10

3 Algorithms Explanation 12
- 3.1 Shift Cipher 12
- 3.2 Permutation Cipher 12
- 3.3 Vigenère Cipher 13
- 3.4 Affine Cipher 13
- 3.5 Substitution Cipher 14
- 3.6 Hill Cipher 15
- 3.7 Block Cipher 18
  - 3.7.1 Counter mode 18
  - 3.7.2 ECB mode 19
  - 3.7.3 CBC mode 19
- 3.8 ElGamal Cipher 20
- 3.9 ECC Cipher (Elliptic curve) 20
- 3.10 RSA Cipher 21
- 3.11 Rabin Cipher 21

4 Cryptosystem performances 23
- 4.1 Peak Signal-to-Noise Ratio (PSNR) 23
- 4.2 Classic Cryptosystems 23
  - 4.2.1 Shift Cipher 23
  - 4.2.2 Vigenère Cipher 23
  - 4.2.3 Image Ciphers 24

1 Introduction

This document serves as an instruction book for the web application Crypto-Box, developped as a proyect for the course Introduction to Cryptography.
CryptoBox is an app that allows the user to employ methods taught in the course to encrypt or decrypt messages or images. The decryption can either be done with use of a key, as the destinatary of the encrypted message would, or in some cases, depending on the method used to encrypt the message, the user may try to decrypt or facilitate the decription of the message without a key, as an interceptor would.
Most sub-sections in this manual will be focused on a particular method. The first section will show how to use the functions of the app related to a method, and explain what the inputs and outputs should be.
In a second section, the algorithms will also be described and explained. That is, the way they encrypt, decrypt and depending on the cryptosystem, attack a crypted text.
Finally, an analysis of the performance of these algorithms will be made. Things such as execution duration, accuracy and effectiveness will be taken to consideration. Of course, all these crypting and decrypting algorithm are deterministic, so the accuracy will only vary for the cryptanalyses. On the other hand, by effectiveness we mean how well were the messages crypted, this will mostly apply to the ones that crypt images, and what will need to be determined is how distinguishable is the crypted image from the original.

2 Instructions for use

2.1 Home Page

When loaded, the application will display this page. On the top part of it, the user may deploy "Classic Systems", "Block Cipher" or "Public Key", to choose one of the contained cryptosystems by clicking on it. This last action takes us to a page where we can do everything related with the criptographic system, that is encrypting, decrypting, doing a cryptanalysis, or generating a key.

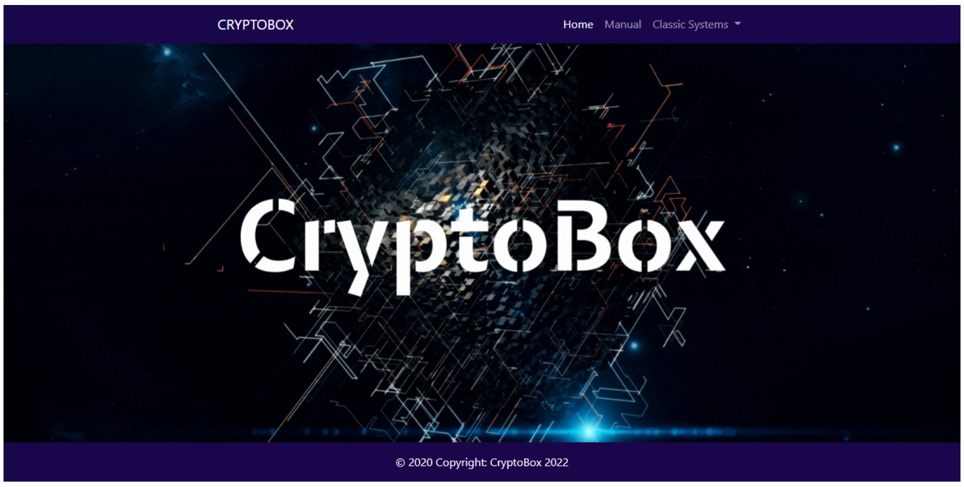


Figure 1: Home Page

2.2 Shift Cipher

- **Encryption:**

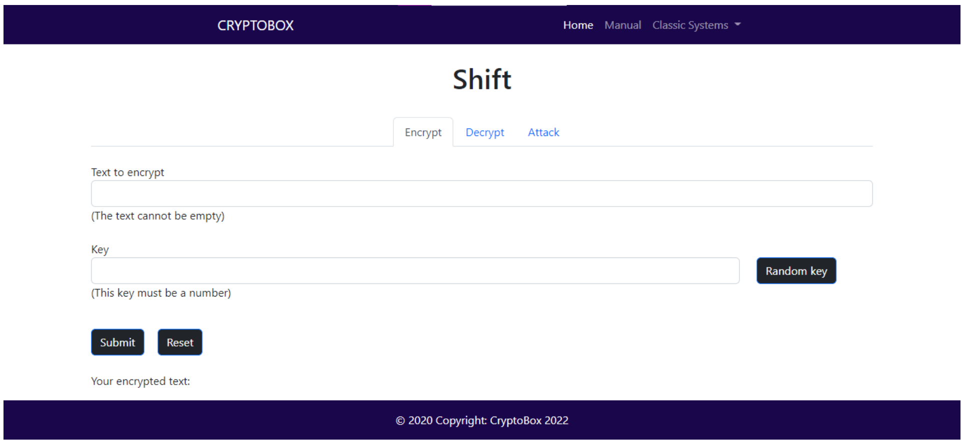
  Simply type the text you want to crypt, then input a key, that can be any number between 1 and 25. To see the crypted text confirm by clicking on the button "Submit".

- **Decryption:**

  Copy or type a text that has been crypted with this method, then input the key and press the button.

- **CryptAnalysis:**

  Again, copy or type a large enough text that has been crypted using this method, then, without need of a key, the decrypted text will be returned.



Shift

Text to encrypt

(The text cannot be empty)

Key

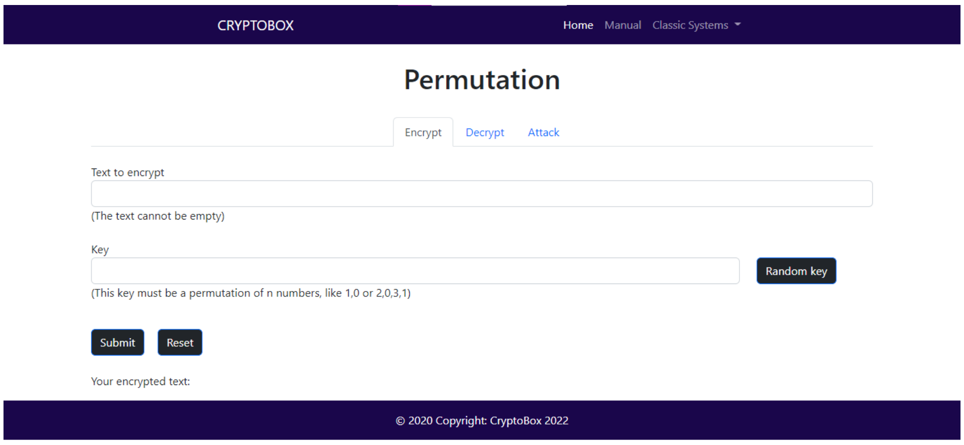
(This key must be a number)

Your encrypted text:

© 2020 Copyright: CryptoBox 2022

Figure 2: Shift Page

2.3 Permutation Cipher



Permutation

Text to encrypt

(The text cannot be empty)

Key

(This key must be a permutation of n numbers, like 1,0 or 2,0,3,1)

Your encrypted text:

© 2020 Copyright: CryptoBox 2022

Figure 3: Permutation Page

- **Encryption:**

  The encryption needs a text and a permutation of a given longitude as an input. The permutation is a chain of suffled increasing consecutive numbers, starting by 0 and separated by commas. A few examples are:
  0,2,1
  2,3,1,0
  3,2,1,0

- **Decryption:**

Just like the encryption, the decryption only needs the encrypted text and the permutation that was given to encrypt it.

- **CryptAnalysis:**

  Only the text is needed to make the cryptanalysis. Thus, after typing the crypted text the user just needs to press the button. The output will be a list of possible decrypted texts.

2.4 Vigenère Cipher

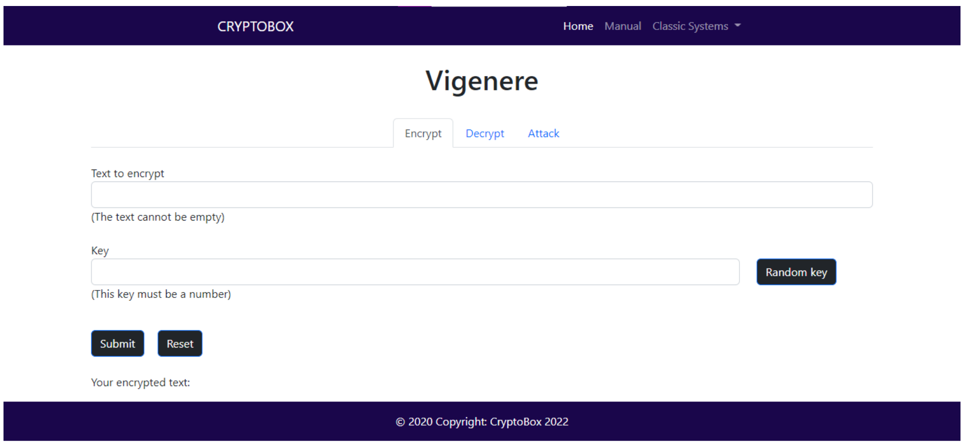


Figure 4: Vigenère Page

- **Encryption:**

  Type the text that is to be crypted, then choose a key and type it in it's corresponding slot. This time the key isn't a number, it needs to be a word, or a chain of characters. Finally, press the button to confirm.

- **Decryption:**

  Again, just copy or type a text that has been cripted with this method, then input the key, that in this case is a chain of characters, and press the button.

- **CryptAnalysis:**

  If you don't have the key to the crypted text, you can input it in the cryptanalisis option, press the button to confirm. The decrypted text will come out as an output.

2.5 Affine Cipher

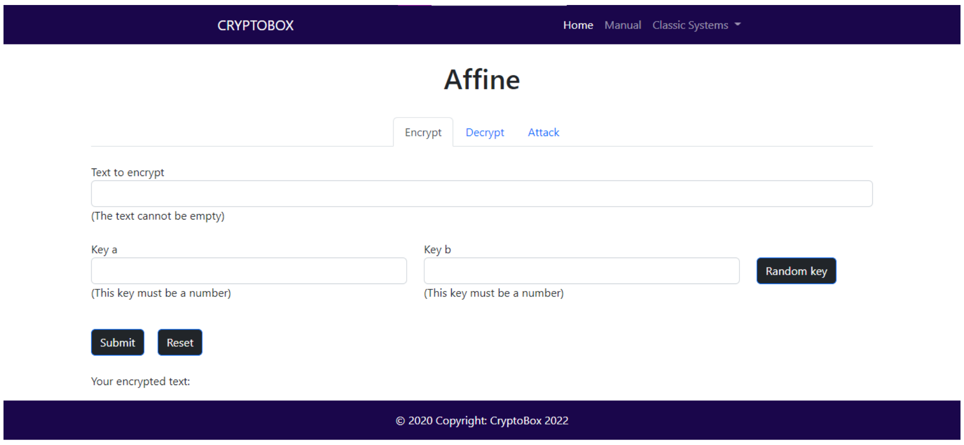


Figure 5: Affine Page

- **Encryption:**

  After typing a text, you'll need to choose a key that consists of two numbers, a and b. b can be whatever number you choose, but a has to be coprime with 26. The crypted text will appear after you confirm by pressing the button.

- **Decryption:**

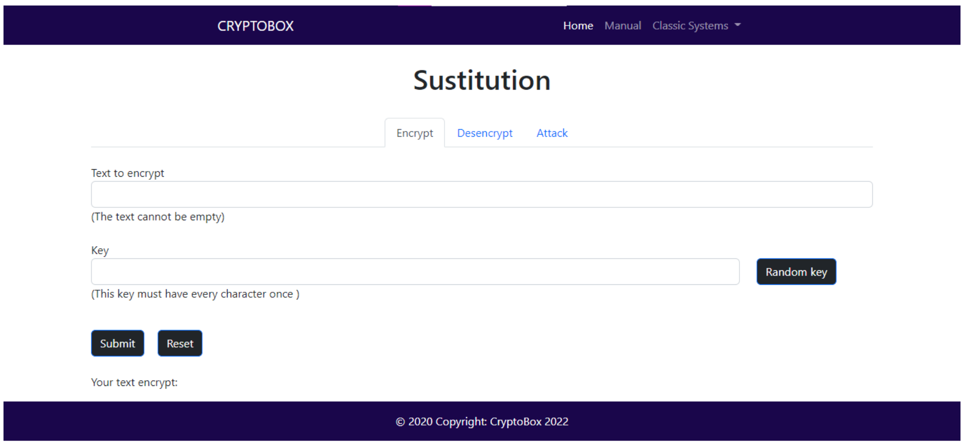
  If you know the values of a and b, type them in their corresponding slots, then type the crypted text and confirm to decrypt.

- **CryptAnalysis:**

  The goal here is to find the key, a and b, using the user's help. In order to do this, the user can input the text and get the frequency of each letter in the crypted text. Using this, and a chart of the average frequency of each letter in the english language, the user can try to guess two pairs of letters that will be used to make a decryption.
  These pairs are made such that the first letter is a letter in the crypted text, and the second a speculation of what letter it will correspond to in the decrypted text. The user should choose these by seeing which have a very similar frequency.
  However, not any two pair of letter can be chosen, as the resulting system of modular equations might not have a solution. If the pairs are (a, b) and (c, d), meaning that when decrypted the as will become bs and the cs will become ds, the following conditions must be met: $c - a$

and $d - b$ have to be coprime with 26.
When such pairs are found, a decryption will be made and the user will be able to determine if it makes sense. If not, he or she can try with another two pairs of letters.

2.6 Substitution Cipher



Sustitution

Text to encrypt

(The text cannot be empty)

Key

(This key must have every character once)

Your text encrypt:

© 2020 Copyright: CryptoBox 2022

Figure 6: Substitution Page

- **Encryption:**

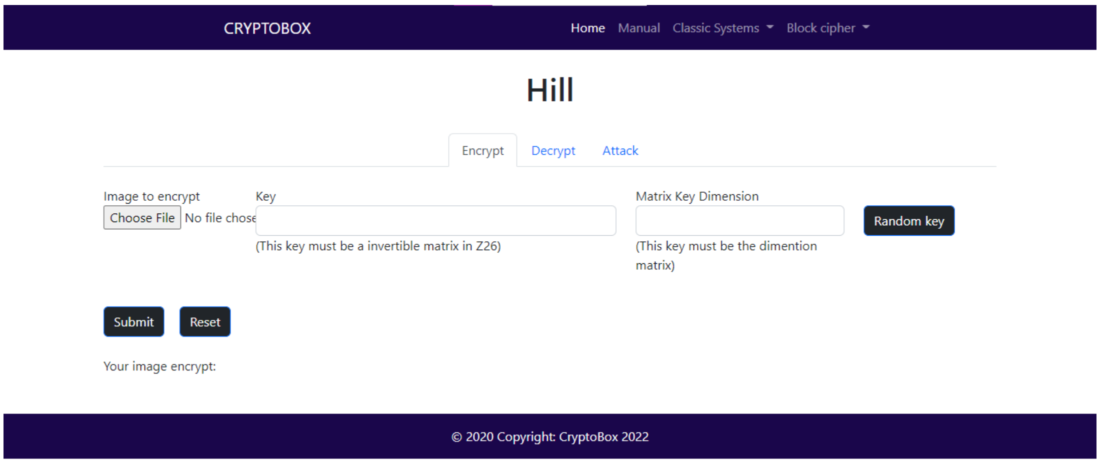
  For this encryption a string, or a “word” of 26 letters is needed. This string mustn’t have any repeated letters, as it will indicate which letters goes to which. If for example the string is pmeqt... , the *a* will be replaced by the *p*, the *b* by the *m*, the *c* by the *e* and so on. When the text and the substitution key are typed in, confirm to get the crypted text.

- **Decryption:**

  Type in the 26 letter string and the crypted text, then press the button to get the decrypted text.

- **CryptAnalisis**

  The possible encryptions of a text using this method are 26!, which is far to great to try to consider all of them, or to implement an efficient analisis.



Hill

Image to encrypt

Key

(This key must be a invertible matrix in Z26)

Matrix Key Dimension

(This key must be the dimention matrix)

Your image encrypt:

© 2020 Copyright: CryptoBox 2022

Figure 7: Hill Page

2.7 Hill Cipher

- **Encryption:**

  Three inputs are necessary in this section, the image, the size of the square matrix that will be the key, and said matrix. After pressing the button, the crypted image will be returned.

- **Decryption:**

  Similarly, the decryption will need the crypted image, the size of the square matrix and the matrix, and the return will be a decrypted image.

- **CryptAnalisis**

  The cryptanalysis works for text, so the user will have to input text and the app will return all possible keys of size 2×2, 3×3 and 4×4.

2.8 Simplified-DES Cipher

- **Encryption:**

  Simply type the text you want to cript, then input a key, that has to be a chain of 1s and 0s of longitude 8, that is, a byte. To see the crypted text confirm by clicking on the button "Submit".

- **Decryption:**

  Copy or type a text that has been cripted with this method, then input the key and press the button.

2.9 Triple Des Cipher

- **Encryption:**

  This cipher is implemented to encrypt images, so to be used the user will need to upload an image, and a key, which will have to be in the form of a 16 character string that will be later on transformed to bytes. Aditionally, the user may have to input an initializing vector, in case they want to use the CBC, OFB or CFB mode, or a counter if they wish to use CTR mode. These two will also be taken as strings. After pressing the button, the crypted image will be returned.

- **Decryption:**

  The decryption will need the same inputs, except the images that now needs to be a crypted one.

2.10 AES Cipher

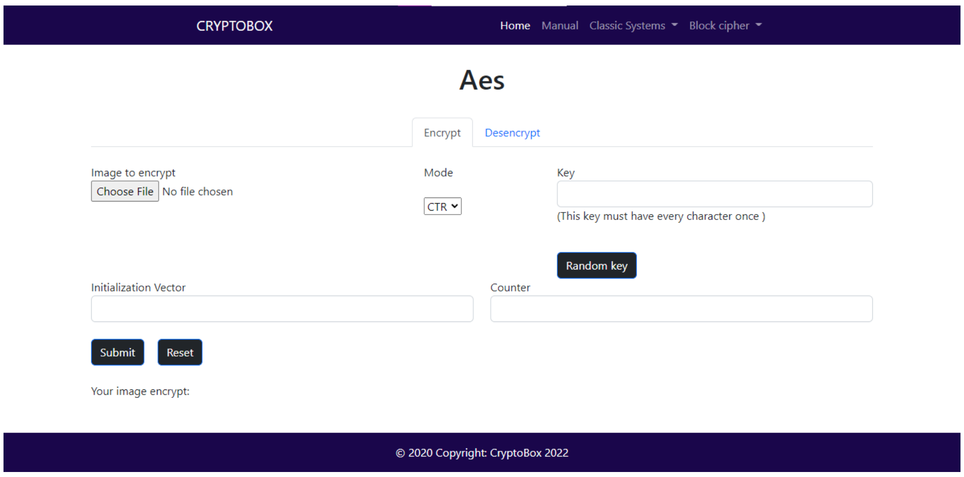
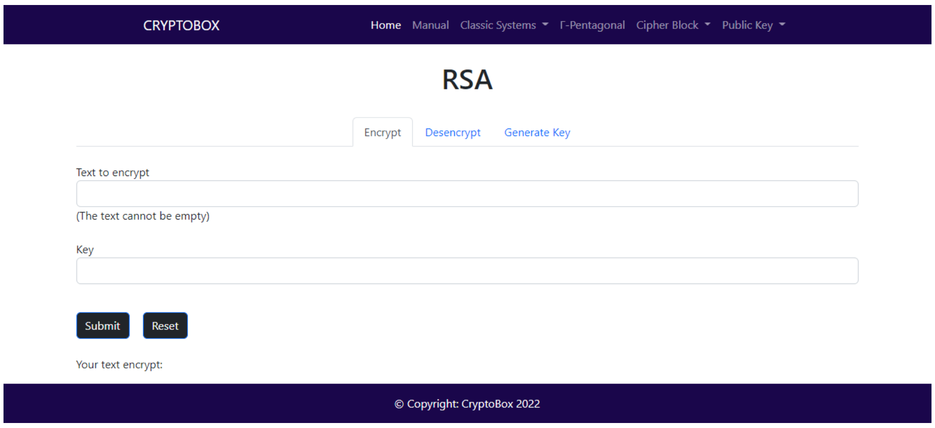


Figure 8: AES Page

- **Encryption:** Four inputs are necessary in this section, the image, the mode of encryption that by default its CTR, the key that could be 16, 24 or 32 characters, the Counter and the Initialization Vector must be exactly 16 characters . After pressing the button, the crypted image will be returned.

- **Decryption:** Similarly, the decryption will need the crypted image, the size of the square matrix and the matrix, and the return will be a decrypted image.

2.11 RSA Cipher



RSA

Text to encrypt

(The text cannot be empty)

Key

Your text encrypt:

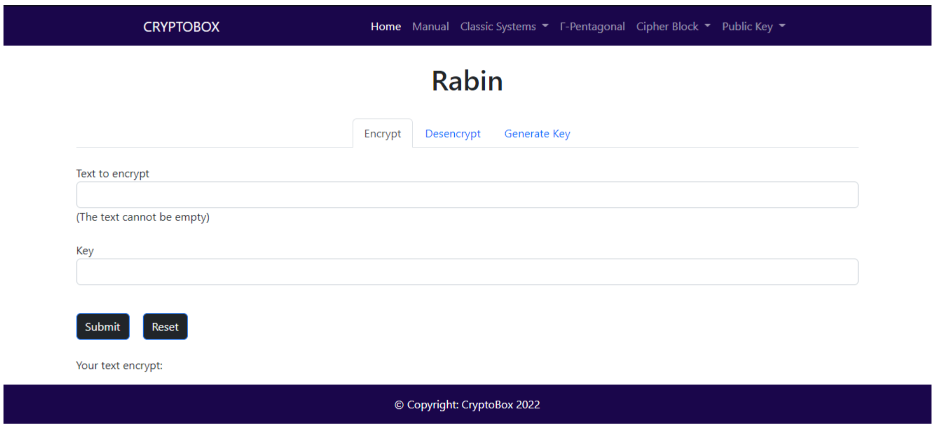
© Copyright: CryptoBox 2022

Figure 9: RSA Page

- **Generate Key:** Here the user may input two different prime numbers, if they confirm, a public and a private key will be returned. If the user presses "random", the slots will automatically be filled.
- **Encryption:** To encrypt, all is needed are the public key and a text. When both slots are filled, the user can confirm to get the crypted text.
- **Decryption:** It's a very similar procedure to decrypt, only this time the private key is needed. By pressing "Decrypt" the user will receive the decrypted text.

2.12 Rabin Cipher

- **Generate Key:** Here the user may input two different prime numbers, if they confirm, the product of the two will be returned, which will be used as a public key.
- **Encryption:** To encrypt, all is needed are the public key, that is the product of two prime numbers, and a text. When the user confirms they will get the crypted text.



Rabin

Text to encrypt

(The text cannot be empty)

Key

Your text encrypt:

© Copyright: CryptoBox 2022

Figure 10: Rabin Page

- **Decryption:**

  Again, similar to the last cryptosystem, the private key is needed. By pressing "Decrypt" the user will receive the decrypted text.

3 Algorithms Explanation

3.1 Shift Cipher

- **Encryption:**

  This cipher is the most simple. We simply "rotate" the letters of the alphabet how ever many times the key indicates us to. So, if the key is for example 1, the letter a will become the letter b, the letter b will become the letter c and so on.

- **Decryption:**

  To decrypt we can simply rotate back the same numbers of times, or complete the rotation, by contiuing to rotate the necessary number of times.

- **CryptAnalysis:**

  This algorithm makes use of a constant that every language has, which is the sum of the squared frecuency that each letter has to show up. English's constant is around 0.65, so we can take a text, calculate it's value for this function and determine wether it's likely to be english. We do this by multiplying the frecuency of each letter in english by the frecuency of a rotated letter, then summing these values. When the rotated letter matches the letter in english, the result should be close to the constant, we'll know theh we found the right "rotation". If we do this for every possible "rotation" of the crypted text, we can find which seems to be the closest to english.

3.2 Permutation Cipher

- **Encryption:**

  To encrypt we need our text and a permutation of 0 to n numbers, for example: $3, 0, 2, 1$, which will be our key. We divide the text into the length of the key and perform the given permutation. If the key does not exactly divide the text, x is added to complete it.

- **Decryption:**

  We need the same as for encrypting and it will do the same process, the difference is that it will do it with the reverse permutation.

- **CryptAnalysis:**

Here we will assume that the key is of length one divisor of the length of the text so for ease, we will only check the possibilities with all permutations of length less than 6 that divide the text. If the length of the text is prime, we take permutations of the same length.

3.3 Vigenère Cipher

- **Encryption:**

  We remind the reader this cypher uses a word for it's encryptions. The algorithm reads the first letter of the key, takes it's asigned value, and "rotates" the first letter of the plain text by said value, as in a shift cypher. Then, the second letter of the key is read and the same is done with the second letter of the text. This is done repeatedly, and if the key runs out of letters, it cycles and uses the first letter again.

- **Decryption:**

  The decryption follows the same principle as the encryption, only this time the rotations are done backwards, thus resulting on the original letters.

- **CryptAnalysis:**

  The cryptanalysis is split in two parts. First, we need to find the length of the key, then we must find the key to finally decrypt the text. To verify if a given length, L, is the right one, split the text in L subtexts, then sum the squares of the frecuency of every letter in the crypted text. If the result is close to 0.65 for each subtext, we know we found the right key length.
  Now all we have to do is separate the text into a number of subtext equal to the length of the key. Note that we can see each of these subtexts as having been crypted with a shift cypher, so all we have to do is a shift cypher cryptanalisis to determine each shift and then rebiuld the key.

3.4 Affine Cipher

- **Encryption:**

  This cypher takes two keys, *a* and *b*. To encrypt, the value assigned to a letter will be multiplied by *a*, and then *b* will be summed. Finally, we take the result mod 26 and give it's assigned letter.

- **Decryption:**

  Contrary to the encryption, first we substract b, and then we find the inverse of a and multiply by it. Note that is why a has to be coprime with 26, if it was'nt it woudln't have an inverse.

- **CryptAnalysis:**

  Here we take two crypted letters and their decrypted counterparts. This way we have a sistem of modular equations, that means, if the letters chosen are adequate, we can solve for a and b.

3.5 Substitution Cipher

- **Encryption:**

  Substitution of single letters separately—simple substitution—can be demonstrated by writing out the alphabet in some order to represent the substitution. This is termed a substitution alphabet. The cipher alphabet may be shifted or reversed (creating the Caesar and Atbash ciphers, respectively) or scrambled in a more complex fashion, in which case it is called a mixed alphabet or deranged alphabet. Traditionally, mixed alphabets may be created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet in the usual order.
  The first letter, A, can be mapped to 26 possibilities (including A itself). The next letter, B, can be mapped to 25 possibilities. The third letter, C, can be mapped to 24 possibilities. And so forth.

  This generates a space of (read the large number in the picture above). That is right. It is more than 88 bits of security or 309485009821345068724781056 possibilities.

- **Decryption:**

  If we have the key we can make an inverse correspondence can be made with the initial key and the positions of each character.

- **CryptAnalysis**

  A frequency analysis attack involves looking at how many times each letter appears in the encrypted message, and using this information to crack the code. A letter that appears many times in a message is far more likely to be "T" than "Z", for example.

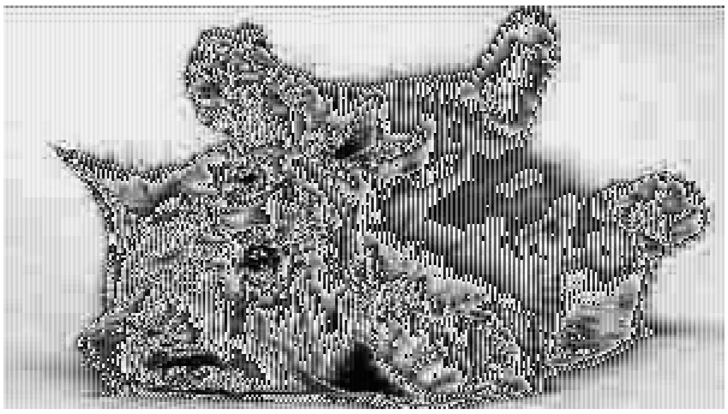
3.6 Hill Cipher

- **Encryption:**

  The user firstly upload the image they desire to encrypt by clicking the **Choose file** button, then the user must enter the key matrix for encryption, this key must be invertible in Z_{256}, e.g 3, 4, 0, 1 represents the 2x2 matrix with entries $(0, 0) = 3$,$(0, 1) = 4$,$(1, 0) = 0$ and $(1, 1) = 1$, also the user must provide the matrix's dimension, in the previous example the matrix's dimension was 2, the hill encryption algorithm work as follows: it takes the image provided by the user, convert it to from RGB to black and white, then it convert the image to it's pixel representation as a matrix, it flatten the matrix, start taking 1xn matrix blocks and multiplying them with the key matrix, and finally the image is encrypted, there is an example:

  Supose we want to encrypt the following image:

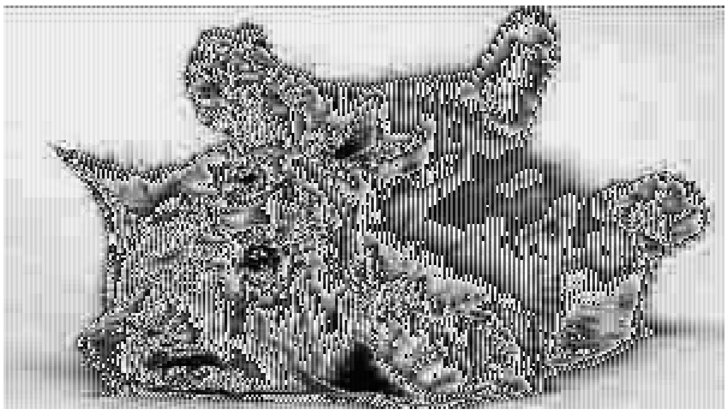
  The encryption would then be:



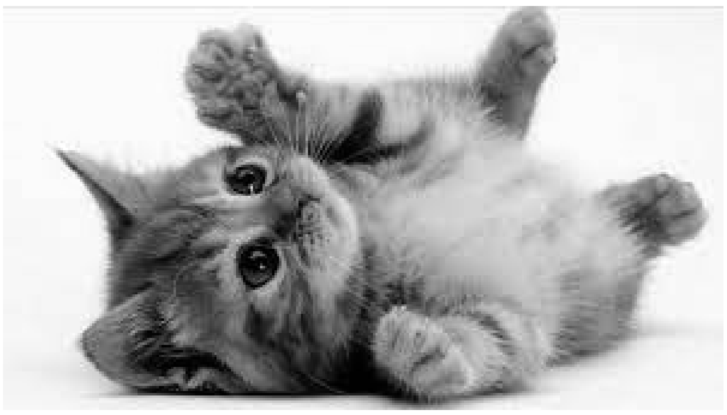
- **Decryption:**

  The user firstly upload the image he/she desires to decrypt by clicking the **Choose file** button, then the user must enter the key matrix for decryption, this key must be invertible in $_{256}$, e.g 3, 4, 0, 1 represents the 2x2 matrix with entries $(0, 0) = 3$,$(0, 1) = 4$,$(1, 0) = 0$ and $(1, 1) = 1$, also the user must provide the matrix's dimension, in the previous example the matrix's dimension was 2, the hill decryption algorithm work as follows:it takes the encrypted image provided by the user, then it convert the image to it's pixel representation as a matrix, it flatten the matrix, start taking 1x*n* matrix blocks and multiplying them with the key matrix inverse (this is why is neccesary the key to have inverse), and finally the image is decrypted, there is an example:

  Supose we want to decrypt the following image:



  Then we get the following as the decryption:



- **CryptAnalisis:**

  Suppose the user has an encrypted message and a plain text, e.g, the encrypted message is **AGGHJKAGBNM**
  and the plain text is **ATTACK**,so it is neccesary that the user have a suspicion about the plain text and we can suppose the key matrix is 2x2, so we suppose $e_k(A,T) = (A,G)$,$e_k(T,A) = (G,H)$,$e_k(C,K) = (J,K)$, we can form the matrix:

$$\begin{bmatrix} A & T \\ T & A \end{bmatrix} = \begin{bmatrix} A & G \\ G & H \end{bmatrix}$$

  If k is the key matrix, we know that:

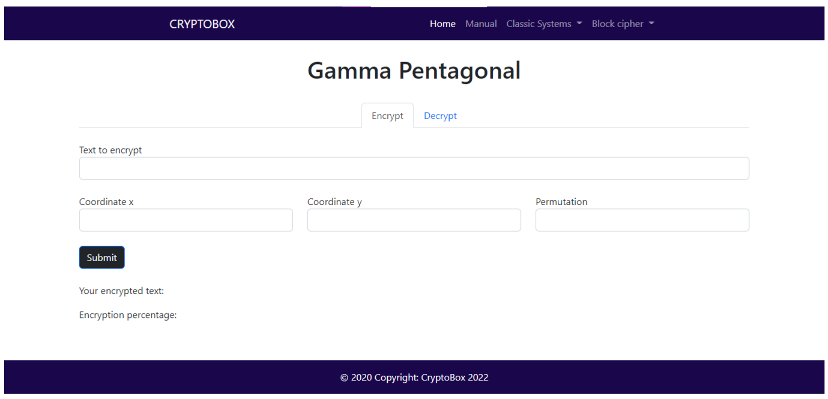
$$\begin{bmatrix} A & T \\ T & A \end{bmatrix} \cdot \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix} = \begin{bmatrix} A & G \\ G & H \end{bmatrix}$$

  We wish to find the key matrix, therefore

$$\begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix} = \begin{bmatrix} A & G \\ G & H \end{bmatrix} \cdot \begin{bmatrix} A & T \\ T & A \end{bmatrix}^{-1}$$

  As we cannot be certain if this matrix has an inverse in z_{256}, we will need to start taking blocks of the key matrix size from the encrypted text and start over again.

- **Gamma Pentagonal** Γ



Gamma Pentagonal

Text to encrypt

Coordinate x

Coordinate y

Permutation

Your encrypted text:

Encryption percentage:

© 2020 Copyright: CryptoBox 2022

Firstly the user must click the **Block Cipher** section, then select **Gamma Pentagonal**.

The user must provide two integers x and y , these two integers will form the origin point (x, y) of the graph.Then the user must provide the permutation for the encryption,e.g,0235814697
, finally the user must give the text for encryption and click the **Submit button**, after few seconds the graph will appear showing the graph with the trayectories , the encrypted text and the percentage of encryption.

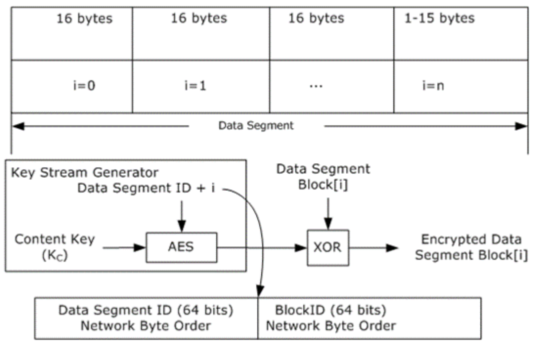
For decryption the user needs to click the **Decrypt section**, the user must provide the encrypted text (trayectories), the origin point and the permutation that were used during encryption and click decrypt, the system will give the user the decrypted text.

3.7 Block Cipher

3.7.1 Counter mode

In counter mode the encryption is applied to separate segments of content data. The definitions for data segment, data segment ID, and block ID MAY vary for different content types and are further discussed in later sections.

Each data segment MUST be encrypted using AES in counter mode . The following diagram illustrates the procedure for encrypting a data segment using this technique.



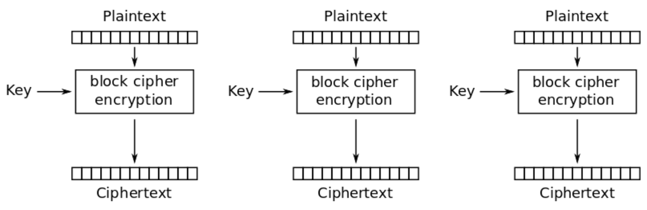
AES in counter mode creates a stream of bytes that MUST be XOR'd with the clear text bytes of the content to create the encrypted content. The key stream generator MUST use an AES round to generate 16-byte blocks of key stream at a time. The inputs to the AES round MUST be the content encryption key (KC) and the 128-bit concatenation of a data segment ID and the block ID within the data segment.

The output of the key stream generator MUST be XOR’d byte by byte with the data from the corresponding block (i) of the data segment. In case the data segment is not evenly divisible by 16 bytes, only the remaining bytes of the data segment from the last block MUST be XOR’d with the key stream and retained for the encrypted data segment.

The data segment ID values MUST be unique for a given KC.

3.7.2 ECB mode

The simplest (and not to be used anymore) of the encryption modes is the electronic codebook (ECB) mode (named after conventional physical codebooks). The message is divided into blocks, and each block is encrypted separately.
The disadvantage of this method is a lack of diffusion. Because ECB encrypts identical plaintext blocks into identical ciphertext blocks, it does not hide data patterns well. ECB is not recommended for use in cryptographic protocols.



Electronic Codebook (ECB) mode encryption

ECB mode can also make protocols without integrity protection even more susceptible to replay attacks, since each block gets decrypted in exactly the same way

3.7.3 CBC mode

In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.
If the first block has index 1, the mathematical formula for CBC encryption is:

$$Ci = EK(PiCi1), C_i = E_K(P_i \oplus C_{i-1}), C0 = IV, C_0 = IV,$$

while the mathematical formula for CBC decryption is

$$Pi = DK(Ci)Ci1, P_i = D_K(C_i) \oplus C_{i-1}, C0 = IV.C_0 = IV.$$

3.8 ElGamal Cipher

- **Key Generation:**

  We first choose an prime p for which the discrete logarithm problem is difficult to compute (non-tractable) in Z, in order to form a generated abelian group Z_p and elements $g, a \in Z_p$ finally we compute $h \equiv g^a mdp$,such that the public key = $\{(p, g, h\}$ and the private key as a

- **Encryption:**

  The sender selects an element $k \in Z_p$ and computes $r = g^k modp$ and $s = h^k = g^{ak} modp$, let the Message be M and the final encription is $(r, M * s) = (g^k, M * s)$, for efficient transmission of text instead of just numbers, characters are passed one by one in ascii utf-8 format.

- **Decryption:**

  The receptor catch $(r, M * s)$ and computes $s' = r^a modp = g^{ak} modp$, dividing $M * s)/s$ it follows that the clear text is retained, and matched every single output with it corresponging character

3.9 ECC Cipher (Elliptic curve)

Based on the same principle as ElGamal Cipher, the problem of discrete logaritm non-tractable over Z_p, and having in count that an elliptic curve $y^2 = x^3 + ax + b$ and their asociate solution set $\in_p x_p$ to the equivalence

$$y^2 \equiv x^3 + ax + b(mdp)$$

- **Key Generation:**

  Having in mind we should choose a big prime number to be able of encrypt larger range of character (as utf-8) because

  $$p + 1\sqrt{p}|E|p + 1 + \sqrt{p}.$$

- **Encryption :** In particular we find one solution by replacing and determining which of the values obtained are quadratic residuals mod p, by the operation + : E X E $\rightarrow$ E, closed to E, if P = (x1, y1), Q = (x2, y2) E

  $$x3 =^2 x1x2 = \frac{y2y1}{x2x1}, ifP \neq Q$$
  $$y3 = (x1x3)y1 = \frac{3x^21+a}{2y1}, ifP = Q,$$

  so we use as public key p prime,a, b as the coeficient of the elliptic curve and a random k $\in N$

- **Decryption :**

  Let be the message M and the private key key, The receptor catch (X_1, Y_1) a point on the curve , and computes ${}_xM = X_1 - Y_1^{key} mod p$ it follows that every point correspond to a unique character in the hash of characters-integers, for that M is retained, and matched every single output with it corresponging character

3.10 RSA Cipher

- **Key Generation:**

  With two different prime numbers, p and q, the public and private key are generated. The public key is made of e and n, n being the product between p and q, and e being a random number coprime such $e < \phi(n)$ and e is coprime with $\phi(n)$.
  The private key on the other hand is made up of d and n, being d the multiplicative inverse of e module $\phi(n)$.

- **Encryption:**

  For this cryptosystem every character of the plain text will be crypted separetely. Let's say we asign the value k to a certain character. Then, it's crypted value will be equal to k^e mod(n). By switching the crypted data to base 64, we can print a crypted text.

- **Decryption:**

  After reverting the convertion to base 64, we take each character, and it's assigned decrypted-value, will be c^d mod(n).

3.11 Rabin Cipher

- **Key Generation:**

  The public and private key are generated with two different prime numbers, p and q. The public key is $d = p \times q$, and the private key will consist of p and q.

- **Encryption:**

  Every character of the plain text will be crypted separetely again. Let's say we asign the value k to a certain character. This time, we'll convert k to binary, and duplicate it, to get a new value, k_1. If for example `bin`(k) = 1001, `bin`(k_1) = 10011001. This operation will come in handy for the decryption. The assigned crypted value of k_1 will be equal to

$k_1^2 \bmod(n)$. By switching the crypted data to base 64, we can print a crypted text.

- **Decryption:**

  After reverting the convertion to base 64, we take each character and decrypt it separately. Let us call a given character C. First, we need to calculate a and b, such that $a \times p + b \times q = 1$. Then we compute r and s, using the following formula:

$$r = C^{\frac{p+1}{4}} \bmod(p)$$
$$s = C^{\frac{q+1}{4}} \bmod(q)$$

  Finally, we determine X and Y such that:

$$X = a \times p \times r + b \times q \times s \bmod(p)$$
$$Y = a \times p \times r - b \times q \times s \bmod(q)$$

  -X, X, Y and -Y then result in all posible decryptions. However, we know the true one is duplicated, that means it's right half is equal to it's left's. having determined which is the true one, we return it to it's original state, by removing the duplicated part of it's binary, and return it as the decrypted key.

4 Cryptosystem performances

In this section we'll talk about the effectiveness of the cryptosystems in the app, by giving average execution times for the more complex algorithms that don't run instantly, and for the ones that use images, we'll determine how well the image was crypted for given examples, making use of the PSNR criteria. We'll also provide information about the accuracy of the attacks worth mentioning. That is, those that intead of using brute force, try to guess the original text by analysing frequencies.

4.1 Peak Signal-to-Noise Ratio (PSNR)

We will use PSNR for measuring how good images encryptation is, PSNR is the ratio between the maximum possible power of an image and the power of corrupting noise that affects the quality of its representation. To estimate the PSNR of an image, it is necessary to compare that image to an ideal clean image with the maximum possible power.

$$PSNR = 20log_{10}(\frac{L-1}{MSE})$$

Here, L is the number of maximum possible intensity levels (minimum intensity level suppose to be 0) in an image and MSE is the usual mean squared error.

4.2 Classic Cryptosystems

4.2.1 Shift Cipher

As this cryptsystem is fairly simple, the encryption and decryption will always be successful and will be returned virtually immediately. However, the success of the cryptanalisis will depend on the length of the crypted text, for it uses an attack based on frequencies, which become less trustworthy when is database is small. As testing suggests, when the clear text is at least 15 characters long, the attack will be accurate. If it is between 10 and 15 characters long, there's around a 80% chance the attack will be succesful. If it's even shorter than that, the attack will almost surely fail.

4.2.2 Vigenère Cipher

This cryptosystem's attack is pretty similar to the Shift's, so a proportional quantity of successes is to be expected. The success however will this time depend on the length of the key as well as the text's. Analogous to Shift, the text should be 15 times longer than the key, due to the cryptanalisys having to attack *n* crypted by Shift texts, being *n* the length of the key.

4.2.3 Image Ciphers

As for the cryptosystems falling into this category, the execution was virtually immediate, even for large images. This is due to the parallel execution of encryptions in the implementation that was used. However, this may vary depending on the user's internet connection's quality. About the cruptoom quality, when the Psnr value was calculated, the results were very similar, even for different cryptosystems. We noted it was always very close to 29.5 db, which is understandable for encrypted images that have the borders of figures still showing.