A Study of Simple Cryptographic Methods Using Matrix Operations

Dinesh Nariani, Alen Kuriakose, Riddhi Parekh,
Abhiram Srikanth, Samasti Bhatt

Mathematical and Physical Sciences Division, School of Arts and Sciences,
Ahmedabad University, Navrangpura, Ahmedabad-380009, India

10th April, 2021

Abstract

Cryptography and associated methods are of immense use in the modern world of internet, privacy, AI and Quantum computing. There are many methods for this use of cryptography some more used than others. There are even dedicated cryptosystems and methods for information concerning national securities and intelligence. One simple area of our exploration is to understand the use of linear algebra-based operations in cryptography and to study it. Finally, we will also develop a program for the same.

I. Introduction

Cryptography at its most basic sense is the science of secret writing. It is the study and practise of developing and adapting methods of transmitting secured data using minimal codes that only allows the receiver of an information to accept the actual information conveyed. The word 'Cryptography' has a Greek origin and is derived from the word 'kryptos' meaning hidden. The earliest known origins of cryptic texts and thus cryptography can be traced back to ancient Egypt, although the sense in which cryptography was used was a bit different. This does not end here; many old civilizations are found to have used cryptography in some primitive forms. Caesar cipher used by Julius Caesar is a notable and historically important one. It is said that modern cryptography originated among the Arabs who introduced permutations and combinations into the method. Later on, many different techniques were developed and the developments jumped up during the world wars, especially the second one. More mathematical and abstract concepts were used for the purpose through the next decades. Today cryptography finds many applications ranging from banking, healthcare and software to internet, cybersecurity and national secrets.

II. The Basics

Cryptography can encrypt and decrypt almost any type of file. We will focus on images and texts which are extensively used in daily life. The aims of cryptography can be identified by the following five primary functions:

1. Privacy – Making sure that the information is not read by anyone other than the intended receiver.
2. Non-repudiation - Any methods to ensure that the sender really sent the information.
3. Integrity/Reliability - Making sure that the information has not been altered in anyway.
4. Authentication - Making sure that the identities are matched.
5. Key-exchange – Methods by which keys required for decryption are shared between sender and receiver.

The raw data from the sender which is un-encrypted is known as the plaintext. This is then encrypted into another form which is known as the ciphertext. These are objects which are sent to the receiver. The receiver receives the encrypted ciphertext which are decrypted using suitable keys which only the receiver has. The types vary according to different varieties of processes used in each step. A flowchart representation of this process is given in figure 1. The process of encryption and decryption involve an important task of generating keys. This is where we will use the concept of Linear Algebra, our prime topic of concern. The concepts of linear transformations, vector spaces, matrices, inversion of matrices, algorithm management etc. are important.

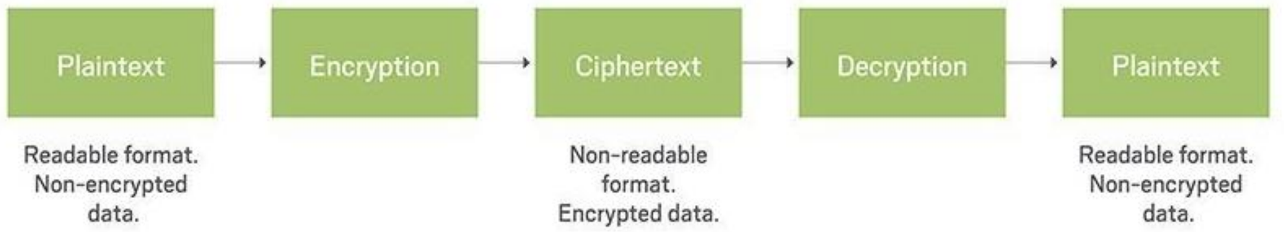

Plaintext
Encryption
Ciphertext
Decryption
Plaintext
Readable format. Non-encrypted data.
Non-readable format. Encrypted data.
Readable format. Non-encrypted data.

Figure 1: Data/Information flow from sender to receiver (Image Credit- https://searchsecurity.techtarget.com/definition/cryptography#:~:text=Cryptography%20is%20a%20method%20of,%22%20stands%20for%20%22writing.%22)

As previously mentioned, the number of ways for categorizing the steps involved are vast. Based on number and types of keys used in encryption and decryption, they can be classified to the following three –

i. Secret-key Cryptography/Symmetric-key Cryptography
ii. Public key Cryptography
iii. Hash Functions

Matrices and their manipulations are an important concept in use here. A matrix is essentially a rectangular array of numbers. They are a way of representing numbers for easier calculation and understanding. Thus, matrices form a tool for mathematics. The inverse computation, multiplication computation etc. are used here inevitably. During a key generation process, if we carry out a modulation of prime numbers, we get a lower triangular matrix for encryption. On the contrary, we get an upper triangular matrix for decryption. Note that we can also include other mathematical operations which have inverse operations to increase the security. For example, using Fourier transform along with matrices.

Finally, before moving on the process algorithms, we have to define certain elements described above.

i. Inverse of a Matrix – If we have any number (say 5) and we multiply by its inverse (here 1/5), we get 1. This basic concept of numbers is also applicable to matrices. But here we have to be a little more careful, as inverse exists only for square matrix and the determinant should be non-zero. Generally, we can say that a matrix B is the inverse of A (both having n x n dimensions), if

$$AB = BA = I_n$$

ii. Determinant - A determinant is a scalar value associated with a matrix that help in characterising certain properties of the matrix. The determinant can be defined in multiple ways. For a 2x2 matrix its defined as follows

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

iii. Linear Transformation – A transformation T: $V^n \rightarrow V^m$ if it satisfies the following

$$T\ (u + v) = T(u) + T(v)$$
$$T\ (cv) = c.T(v)$$

where V represents a vector space u and v are vectors in V^n and c is a scalar in V.

iv. Modular Arithmetic – congruence modulo method – We say that *a* is congruent to *b* mod *n*, if *n* divides (*a* - *b*). This is written as

$$a \equiv b \bmod n$$

III. The Process

Text

The text required for cryptography is converted to matrix, then transformed to another one using a key matrix and sent to the receiver who receives it and using the key decrypts the information. The step-by-step algorithms are given below.

- Encryption –
    a) Initially, a hash table is made. This consists of numbering characters in a text randomly. This can also be done for all the alphabets in a language and then using the required ones from those. It is made sure that the numbers used are unique and not repeated.

b) The text is then broken down into multiple parts. These parts have fixed length and is essential to be so here. Let us consider it to be divided into ‘n’ parts.

c) After this, an approval is required from senders and receivers’ part. On approval, a matrix is generated. This is the key matrix (K) and is an invertible matrix. The dimension of matrix will be n x n.

d) This is followed by the conversion of each of the n parts to the associated numbers we choose for making the hash table. Each of the parts will form an n x 1 matrix.

e) Next, these vectors are transformed. The process uses the key matrix generated in step (c). Congruence modulo method is then used to find total number of characters.

$$T_a : V_1 \rightarrow V_2$$

$$T_a \equiv K \text{ x mod } n$$

f) This generates a set of numbers (non-negative always) which are then transformed back to normal characters using the same hash table.

g) The text now in hand is the encrypted text or cipher text. This text is then transmitted to the receiver in whichever electronic medium preferred.

- Decryption –

a) The receiver receives the information – the cipher text. The key and hash table are already known to the receiver. Thus, the process of decryption starts at the same moment the information is received.

b) The encrypted text is then divided into ‘n’ part of n x 1 vectors. The same hash table is used for this process.

c) After this, the inverse of the key matrix is found (K^{-1}).

d) Then, using congruence modulo method and the inverse key, the original vectors of the matrix are found.

$$T_b : V_2 \rightarrow V_1$$

$$T_b \equiv K^{-1} \text{ x mod } n$$

e) Finally, the hash table is used once again to convert the numbers back to original text, i.e., the information which was sent to the receiver. This is the last step involved in textual decryption and the process is complete.

Image

The process used for the image cryptography are similar method wise. Here we use the RGB pixel data and also transform size of matrices. The three-dimensional vector space is transformed to a two-dimensional RGB image. As previously key generations are involved and are made sure they are unique and no patterns exist which makes them exploitable.

- Encryption –
    a) The image to be encrypted is chosen and converted to three-dimensional RGB pixel data. This makes it into a form which is easier to use and manipulate
    b) Then, the image is transformed and represented into a two-dimensional matrix (P) of the size m x n. Linear transformations of matrix is the exact process which happens in this step.
    c) After this, a random m x m matrix is generated. This acts as the key matrix (K) for the process. The P matrix is then multiplied with the K matrix to get the encrypted matrix.
    d) The encrypted matrix is then transformed back to the three-dimensional matrix with encrypted RGB pixel data.
    e) This is then sent to the receiver.

- Decryption –
    a) The receiver receives the information – the encrypted 3D image.
    b) The matrix is transformed to the two-dimensional version. This is followed by the key verification for the two-dimensional matrix.
    c) The inverse key is found (K^{-1})
    d) The inverse key is then multiplied with the two-dimensional matrix to get the original two-dimensional decrypted matrix P.
    e) Finally, matrix P, which is two-dimensional is converted back to the original three-dimensional image with RGB data using the same conversion techniques. This is the last step involved and we get the original image and the process is complete.

The time required for encryption and decryption and the MSE (Mean Squared Error) are also interesting topics which follows this process. But we are not discussing it, as it is beyond the scope here.

IV. The Code

The program used for coding was python.

Text

```
# Text Encryption and Decryption
import numpy as np         #Install and import numpy
import random
import secrets
import string


# Function to encrypt string s using key (Password)
def encrypt(s, password):
    n = len(s)


    # Generate plain text matrix
    plain_text = np.arange(n).reshape(n, 1)
    for i in range(0, n):
        plain_text[i][0] = ord(s[i]) - ord('a')


    # Generate key matrix
    key = np.arange(n * n).reshape(n, n)
    for i in range(0, n):
        for j in range(0, n):
            key[i][j] = ord(password[(n * i) + j]) - ord('a')
    # encrypted matrix

    encrypted_text = key.dot(plain_text)

    # generate encrypted string
    ans = ""
    for i in range(0, n):
        ans += chr((encrypted_text[i] % 26) + ord('a'))

    return ans


# Function to decrypt string s using key (Password)
def decrypt(s, password):
    n = len(s)
    print(len(password))

    # generate encrypted matrix
    encrypted_text = np.arange(n).reshape(n, 1)
    for i in range(0, n):
        encrypted_text[i][0] = ord(s[i]) - ord('a')

    # generate key matrix
    key = np.arange(n * n).reshape(n, n)
    for i in range(0, n):
        for j in range(0, n):
            key[i][j] = ord(password[(n * i) + j]) - ord('a')

    key_inverse=np.linalg.inv(key)
    #inverse_key=Matrix(key).inv_mod(26)
    # np.inverse_key = np.array(inverse_key)
    plain_text = key.dot(encrypted_text)

    # generate plaintext string
    ans = ""
    for i in range(0, n):
        ans += chr((plain_text[i] % 26) + ord('a'))

    return ans

# get choice
print("Enter 1 for encryption")
print("Enter 2 for decryption")
a = int(input())

if (a == 1):
    # Encryption BEGINS
    print("Enter string for encrypton(lowercase alphabtes):")
    s = str(input())                              #String to be encrypted
    n = len(s)

    password = ''.join(secrets.choice(string.ascii_lowercase)
                                              for i in range(0,n*n))
    print("Your password is: "+password)
    if (len(password) != n * n):
        print("Invalid length of string/password.")     #Error if password is not n*n letter long
        print("Password should be " + str(n * n) + " letters long")


    else:
        encrypted_string = encrypt(s, password)     #Calling the function to perform encryption
        print("The encrypted string is: " + encrypted_string)


if (a == 2):
    # Decryption BEGINS
    print("Enter string for decryption:")                         #String to be decrypted
    s = str(input())
    n = len(s)
    print("Enter password: ")                                     #Key/Password
    password = str(input())

    if (len(password) != n * n):
        print("Invalid length of string.Please try again")        #Error if password is not n*n letter long


    else:
        decrypted_string = decrypt(s, password)                    #Calling the function to perform decryption
        print("The decrypted string is " + decrypted_string)
```

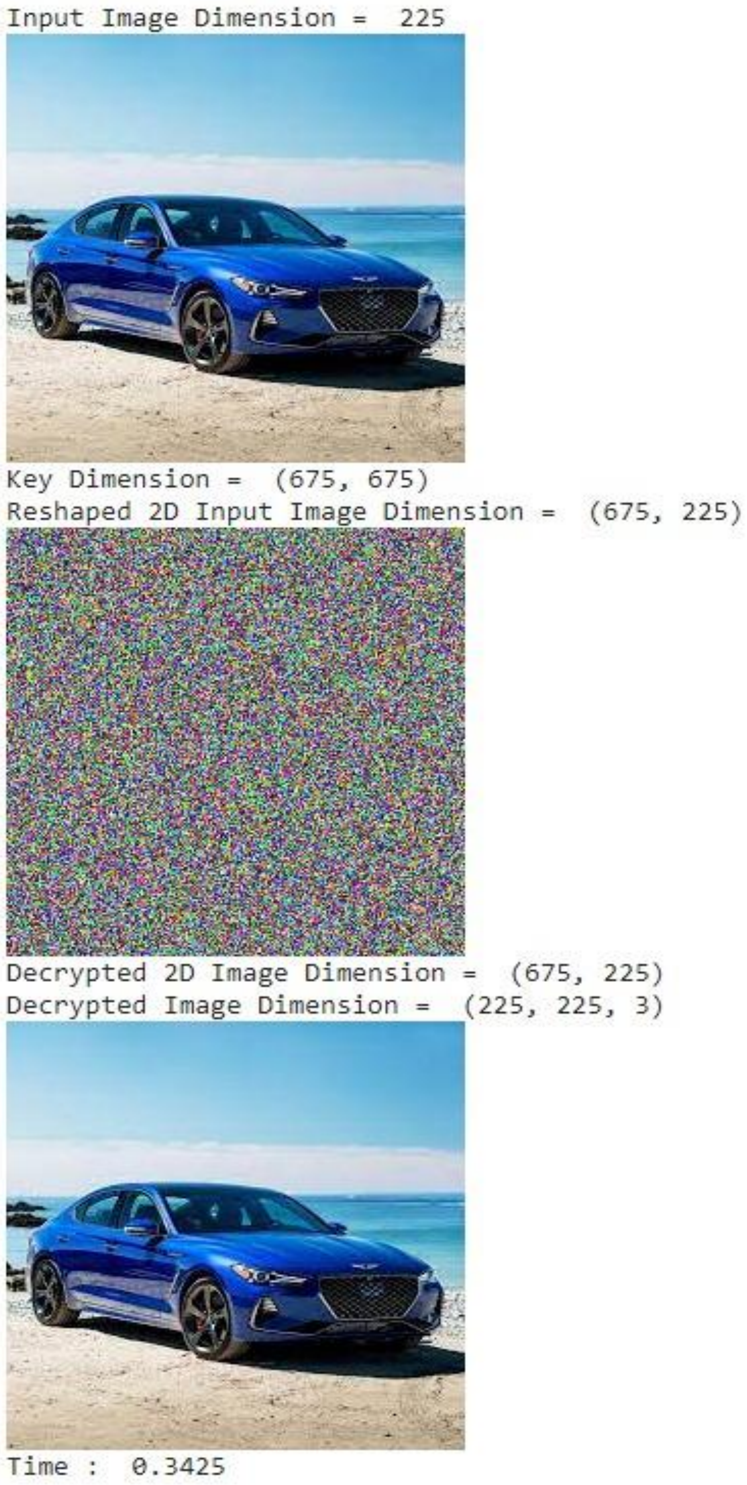
Image

```
# Image Encryption and Decryption
from PIL import Image
import numpy as np
import math
from IPython.display import display
import time


#reading the image into a 3d array
#ROW (height) x COLUMN(Width) x COLORS(3)
src_image = np.array(Image.open('download.jfif').convert("RGB"))
width=src_image.shape[1]
height=src_image.shape[0]
print('Input Image Dimension = ',src_image.shape[1])
display(Image.open('download.jfif'))
#input source



# start time
start = time.time()


#generating a key for encryption and decryption
key= np.random.randint(2, size=(height*3, height*3))
print('Key Dimension = ',key.shape)



#original 3d array of image is now reshaped into 2d
```

```
reshaped=src_image.reshape(height*3,width)
print('Reshaped 2D Input Image Dimension = ',reshaped.shape)

#encrypt_2d is a result of multiplying key matrix(randomly generated) and the reshaped input array
encrypt_2d=key.dot(reshaped)

#the true encrypted matrix , corresponds to encrypted image
encrypt=encrypt_2d.reshape(height,width,3)

#save the image
save_image=Image.fromarray((encrypt * 255).astype(np.uint8))
save_image.save('m_encrypt.png')
display(save_image)

#encrypted image saved as m_encrypt.png

# Decrypt the image
#reshape the encrypted matrix to 2D array/2D matrix
#kinda reverse engineering

decrypt_2d=encrypt.reshape(height*3,width)
print('Decrypted 2D Image Dimension = ',reshaped.shape)

#take inverse of the initially generated key matrix
key_inverse=np.linalg.inv(key)
```

```
#multiply the inverse of the key to reshaped encrypted matrix
#again a transformation , think about the key matrix like a function
decrypted_2d=key_inverse.dot(decrypt_2d)

decrypted=decrypted_2d.reshape(height,width,3)
print('Decrypted Image Dimension = ',decrypted.shape)

# end time
end = time.time()

#final touch up
decrypted=decrypted.astype(np.uint8)

# save the decrypted image
save_image=Image.fromarray(decrypted)
save_image.save('m_decrypt.png')
display(Image.open('m_decrypt.png'))
#decryption done

print('Time : ',round(end - start,4))
```

The following figure (figure 2) represents a result from image cryptography process. First one is the original matrix, followed by the encrypted one and lastly the decrypted one.



```
Input Image Dimension =  225
Key Dimension =  (675, 675)
Reshaped 2D Input Image Dimension =  (675, 225)
Decrypted 2D Image Dimension =  (675, 225)
Decrypted Image Dimension =  (225, 225, 3)
Time :  0.3425
```

Figure 2: Result of image encryption process

V. Conclusion

The project was successfully completed and the results were up to the expectations. The process used here is a very simple demonstration of a highly complex process that is used in our everyday life, most of the time in a level unnoticed by us. In the modern world of internet and AI, cryptography is an essential. Information and its privacy are a matter of concern to levels ranging from personal to national. The methods we explored form a basis for certain kind of cryptography used today. This is obvious, as the same kind of techniques cannot be used for all purposes. However, the basic idea is conveyed. With the advent of quantum technologies and quantum computers, experts are worried about the security of these classical encryptions. Alternative cryptography systems which are based on quantum cryptography are proposed and currently an area of huge interest. Along with many other structures which are changing now, cryptography and thus secrecy is expected to be evolved in the coming decades.

VI. References

1. Biggs, N., SpringerLink (Online service), & Library Genesis. (2008). Codes [electronic resource] : an introduction to information communication and cryptography. In *Internet Archive*. London : Springer. https://archive.org/details/codesintroductio00bigg_911
2. *Chapter 3. Modular Arithmetic*. (n.d.). Www.doc.ic.ac.uk. https://www.doc.ic.ac.uk/~mrh/330tutor/ch03.html#:~:text=For%20a%20positive%20integer%20n
3. Kessler, G. (2019). *An Overview of Cryptography*. Garykessler.net. https://www.garykessler.net/library/crypto.html
4. L, V., & V, B. (2019). Encryption and Decryption Technique Using Matrix Theory. *Journal of Computational Mathematica*, *3*(2), 1–7. https://doi.org/10.26524/cm49
5. Wikipedia Contributors. (2019, October 23). *Cryptography*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Cryptography
6. Zhang, S., Tian, C., Zhang, H., Yu, J., & Li, F. (2019). Practical and Secure Outsourcing Algorithms of Matrix Operations Based on a Novel Matrix Encryption Method. *Kuscholarworks.ku.edu*, *7*. https://doi.org/10.1109/ACCESS.2019.2913591
7. Zwicke, A. (2003). *Global Information Assurance Certification Paper*. https://www.giac.org/paper/gsec/2604/introduction-modern-cryptosystems/104482