Final Project Report

Kaiwen Li

June 7, 2022

1 Introduction

An invertible matrix A has the property $AA^{-1} = I$, which makes it perfect in cryptography. The key matrix is used to encrypt the messages, and its inverse is used to decrypt the encoded messages.

What's more, orthogonal matrix and graph theory can also be used to encode and decode messages.

In this report, we will be discussing three methods. The first one is hill cipher, which uses invertible matrix. The second method is using the properties of orthogonal matrix. The third one is using graph theory to encrypt message and it provides the strongest security.

2 Hill Cipher

2.1 What is Hill Cipher?

In classical cryptography, the Hill cipher is a polygraphic substitution cipher based on linear algebra[con22].

2.2 Encryption

To simplify this example, we're not considering lowercase and uppercase in this case. Each characrter is represented by a number based on its alphabetical order starting with 0. Then we can derive the following table:

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

To encrypt a message with n characters, we need a $n \times n$ invertible matrix. We just need to multiply the matrix of the message and the invertible matri, against modulus 26.

In the following example, we will use a 3×3 matrix. That means, the text we want to encrypt should also be 3 characters long. If the word has more than 3 characters, then we will need to break it into pieces and each piece has exactly 3 characters.

Consider the message 'ACT', here is the corresponding matrix:

$$\begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix}$$

Pick an invertible matrix A:

$$A = \begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}$$

Encrypt the message:

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix}$$

$$= \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} (mod 26)$$

Therefore, we can get a totally different matrix by multiplying the invertible matrix.

2.3 Decryption

Similar to encryption, we just need to multiply the encrypted matrix with A^{-1} to get the original matrix back:

First, the corresponding A^{-1} is:

$$A^{-1} = \begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix}$$

Decrypt the message:

$$\begin{bmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{bmatrix} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} = \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} (mod 26)$$

Hence we retrieved our original message back.

2.4 Security and complexity

I believe that's the easiest matrix application in cryptography. Therefore, the security level of it is low. For example, if we're using the same matrix to encrypt the message, then the same message will always being encrypted to the same matrix. One potential solution to this is to combine a non-linear operation during the encryption such that it can randomize the final output, because matrix multiplication can provide diffusion.

Time complexity: $O(n^3)$

2.5 Conclusion

Hill cipher is a deterministic encryption scheme. That is to say, it is not safe at some sense. And we can see the time complexity is $O(n^3)$, which is kind of inefficient. It can be further improved to $O(n^{2.81})$ by using strassen algorithm[NMTG19].

3 Orthogonal matrix

3.1 Introduction

In several works about cryptography, matrix is used as an application that a vector of plaintext is transform in a vector of ciphertext and, to decrypt, the reverse of the application is used. So, giving a message M and a encoding matrix A, the encrypted message is giving by $X = A * M$. To decrypt, the reversal matrix is used, so the original message is giving by $M = A^{-1} * X$.

As we discussed in section 2, when your text is long, tt takes a long time to find the inverse of the matrix. So if we can use some properties of orthogonal matrix, we can reduce the time to caluclate the inverse of a matrix.

Orthogonal matrices have several properties that make them interesting to diagonalize and find its reverse. So in this section we will talk about how to encrypt and decrypt a message using orthogonal matrix. Here're some of the properties that will be use and are quicker than other matrix method to decrypt:

- Fast inversion
- Real eigenvalues
- Orthogonal eigenvector

3.2 Method

We let A be an orthogonal matrix. Since A is orthogonal, A is diagonalizable, and it can be written as a product of matrix, one of them is a diagonal matrix.

Hence, let $A = C^{-1}DC$, where C is a base matrix and D is the diagonal matrix. Let λ_i be the eigenvalues of A for $1 \leq i \leq n$ corresponding to eigenvector v_i.

Then, we can derive the exponential of A:

$$e^A = C^{-1}e^D C = C^{-1}\begin{pmatrix} e^{\lambda_1} & & & & \\ & e^{\lambda_2} & & & \\ & & e^{\lambda_3} & & \\ & & & \ddots & \\ & & & & e^{\lambda_n} \end{pmatrix} C$$

Proof:

$$\begin{aligned} e^A &= \sum_n \frac{A^n}{n!} \\ &= \sum_n \frac{(C^{-1}DC)^n}{n!} \\ &= \sum_n \frac{C^{-1}DCC^{-1}\dots C^{-1}DC}{n!} \\ &= C^{-1}(\sum_n \frac{D^n}{n!})C \\ &= C^{-1}e^D C \end{aligned} \tag{1}$$

The exponential of D should be:

$$\begin{aligned} e^D &= \sum_n \frac{D^n}{n!} \\ &= \sum_n \frac{1}{n!}\begin{pmatrix} \lambda_1^n & & & & \\ & \lambda_2^n & & & \\ & & \lambda_3^n & & \\ & & & \ddots & \\ & & & & \lambda_n^n \end{pmatrix} \\ &= \begin{pmatrix} \frac{\lambda_1^n}{n!} & & & & \\ & \frac{\lambda_2^n}{n!} & & & \\ & & \frac{\lambda_3^n}{n!} & & \\ & & & \ddots & \\ & & & & \frac{\lambda_n^n}{n!} \end{pmatrix} \\ &= \begin{pmatrix} e^{\lambda_1} & & & & \\ & e^{\lambda_2} & & & \\ & & e^{\lambda_3} & & \\ & & & \ddots & \\ & & & & e^{\lambda_n} \end{pmatrix} \end{aligned} \tag{2}$$

3.3 Encryption Key Construction

To construct the key(matrix C), we need two matrices:k_{pub} and k_{priv}. Let $C = k_{pub}k_{priv}$

Lemma: If k_{pub} and k_{priv} are orthogonal, then C is also orthogonal.
Proof:

$$\begin{aligned} C * C^T &= (k_{pub}k_{priv})(k_{pub}k_{priv})^T \\ &= (k_{pub}k_{priv})(k_{priv}^T k_{pub}^T) \\ &= k_{pub}(k_{priv}k_{priv}^T)k_{pub}^T \end{aligned}$$

Because k_{pub} and k_{priv} are orthogonal, $k_{priv} * k_{priv}^T = k_{pub} * k_{pub}^T = I$, then:

C * $C^T = k_{pub}(k_{priv}k_{priv}^T)k_{pub}^T = k_{pub}k_{pub}^T = I$

3.4 Application

Let's say we want to encrypt 'CRYP'.

To make the matrix more secure, we introduce P, a permutation matrix:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Let our C be:

$$C = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

3.4.1 Encryption

To encrypt the message, we first need to use each of the number of alphabet matched with each letter as eigenvalues and eigenvector:

$$\begin{aligned} C &: \lambda_1 = 3 \\ R &: \lambda_2 = 18 \\ Y &: \lambda_3 = 25 \\ P &: \lambda_4 = 16 \end{aligned}$$

$$X = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix} = \begin{pmatrix} 3 \\ 18 \\ 25 \\ 16 \end{pmatrix}$$

$$D = \begin{pmatrix} e^{\lambda_1} & 0 & 0 & 0 \\ 0 & e^{\lambda_2} & 0 & 0 \\ 0 & 0 & e^{\lambda_3} & 0 \\ 0 & 0 & 0 & e^{\lambda_4} \end{pmatrix} = \begin{pmatrix} e^3 & 0 & 0 & 0 \\ 0 & e^{18} & 0 & 0 \\ 0 & 0 & e^{25} & 0 \\ 0 & 0 & 0 & e^{16} \end{pmatrix}$$

$$\begin{aligned} Y &= PC^{-1}DCX_1 \\ &= \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} e^3 & 0 & 0 & 0 \\ 0 & e^{18} & 0 & 0 \\ 0 & 0 & e^{25} & 0 \\ 0 & 0 & 0 & e^{16} \end{pmatrix}\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix}\begin{pmatrix} 3 \\ 18 \\ 25 \\ 16 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 21e^{18} + 15e^3 \\ 41e^{16} + 9e^{25} \\ 41e^{16} - 9e^{25} \\ 21e^{18} - 15e^3 \end{pmatrix} \end{aligned}$$

Here we got the encrypted matrix Y and it is more secure because we multiplies the original encrypted matrix with a permutation matrix.

3.4.2 Decryption

To decrypt the message, we need to take the inverse of the matrix:

$$P^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned} T &= P^{-1}CY \\ &= \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix}\begin{pmatrix} 21e^{18} + 15e^3 \\ 41e^{16} + 9e^{25} \\ 41e^{16} - 9e^{25} \\ 21e^{18} - 15e^3 \end{pmatrix} \\ &= \begin{pmatrix} -15e^3 \\ 21e^{18} \\ 9e^{25} \\ 41e^{16} \end{pmatrix} \end{aligned}$$

Hence we get our original message back, which is

$$\begin{pmatrix} 3 \\ 18 \\ 25 \\ 16 \end{pmatrix}$$

Converting it to alphabet, we can get CRYP.

3.4.3 Conclusion

This is a quite interesting algorithm, it uses the fact that since orthogonal matrix are always diagonalizable on R, and the emponential of a diagonal matrix is easy to compute, the exponential of orthogonal matrix can be used to encrypt text messages[San14].

But there's one downside of this algorithm, if $C = k_{pub}K_{priv}$, the decomposition of an orthogonal matrix in several orthogonal matrix is not unique.

4 Graph

4.1 Introduction

Right now we only talked about how to encrypt a message using an invertible matrix or an orthogonal matrix, and now we will be looking at how we can encrypt each character into a graph and how to decrypt it.

In this section, we will be talking about an encryption-decryption algorithm using Euler graphs.

4.2 Graphs we will be using

Euler graph
Incidence matrix
Hamiltonian Graph

4.3 Encryption

The message sent by the user will be encrypted into an euler graph in the following steps. An hamiltonian circuit will be traced out from the encrypted graph and it will also be the key to decrypt the message.

1. Convert each alphabet in the message into uppercase.
2. Take the ASCII values of each uppercase alphabets and convert them into binary format.
3. the binary format of each character will XOR with the binary equivalent of 32.
4. Store the resultant XORed binary euivalent in an array $M[i]$.
5. Count $k =$ the number of $1's$ in the binary equivalent from the array $M[i]$.
6. Form an adjacency matrix $A = (a_{ij})$ with the count k, where a_{ij} denotes the element in the i^{th} row and j^{th} column of the matrix. Teh count represents the number of verices of the simple graph. So the principal diagonal elements of the martrix are 0. This adjacency matrix is symmetrix, so $a_{ij} = a_{ji}$.
7. Count the number of $0's$ following the 1 for every element in the array and put $a_{ij} = a_{ji} =$ number of $0's + 1$. For instance, if 10 is in the array, then the $a_{ji} = 2$, if 1000 is in the array then $a_{ji} = 4$.
8. Repeat the above process untill the hamiltonian circuit tracing reaches the end vertex.
9. Upon reaching the last element of the array as 1 that is the binary stream ends with a 1, then make $a_{1j} = 1$ where $j = k$.

10. If the binary stream ends with a 1 and is followed by L number of $0's$, then put $a_{1k} = L + 1$.

11. The the adjacency matrix is the encoded matrix of the message.

4.4 Decryption

Once we receives the adjacency matrix, we can start decrypt the matrix in the following steps:

1. The elements of the adjacency matrix are stored in a temporary array $Z[p]$.

2. We will traverse and take into consideration either the upper triangular matrix or the lower triangular matrix along the main diagonal. This is because of the symmetric nature of the adjacency matrix

3. To build the binary stream, the elements of the $Z[p]$ are expanded.

4. To get back the original message, the operations used in the encoding system are applied backwards.

4.5 Application

Now let's take a look at a concrete example how it actually works. Let our plain text be ***Graph***. First, we convert each character into uppercase and find out the binary number of the ASCII values of each characters:

Alphabet	ASCII value	binary
G	71	1000111
R	82	1010010
A	65	1000001
P	80	1010000
H	72	1001000

Next, we perform XOR operation using the binary value of the alphabet and 32.

Alphabet	G	R	A	P	H
Binary format	1000111	1010010	1000001	1010000	1001000
Binary format of 32	100000	100000	100000	100000	100000
Result	1100111	1110010	1100001	1110000	1101000

Using encryption algorithm and Euler graph, the adjacency matrix of each alphabet is constructed and shown below:

For the alphabet G, there're five $1's$, hence the graph for G has 5 vertices and its adjacency matrix is of 5×5 order which is given below:

$$G = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 3 & 0 & 0 \\ 1 & 3 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Similarly, we can get the the adjacency matrices for the rest of the alphabets in the following:

$$R = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 3 \\ 2 & 0 & 3 & 0 \end{pmatrix}, A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 5 \\ 1 & 5 & 0 \end{pmatrix}, P = \begin{pmatrix} 1 & 0 & 5 \\ 1 & 0 & 1 \\ 5 & 1 & 0 \end{pmatrix}, H = \begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 2 \\ 4 & 2 & 0 \end{pmatrix}$$

The encrypted matrices will be sent out one by one.

For the first matrix the receiver receives [1 3 1 1 1]

For the second matrix the receiver receives [1 1 3 2]

For the 3rd matrix the he receives [1 5 1]

For the 4th matrix the he receives [1 1 5]

For the 5th matrix the he receives [1 2 4].

If we expand the matrix by replace the numbers in the matrix with their binary format, we can get:

13111 = 1100111

1132 = 1110010

151 = 1100001

115 = 1110000

124 = 1101000

Now if we XOR the value with the binary number of 32, we can get back the original ASCII value of our alphabet:

1100111$\rightarrow$ 1000111 $\rightarrow G$

1110010 $\rightarrow$ 1010010 $\rightarrow R$

1100001 $\rightarrow$ 1000001 $\rightarrow A$

1110000 $\rightarrow$ 1010000 $\rightarrow P$

1101000 $\rightarrow$ 1001000 $\rightarrow H$

Hence, the user decrypts the matrices and gets back its original plain text.

4.6 Security and complexity

Comparing to the above algorithm, it provides a much higher security level. This is because:

1. The plaintext is encrypted by each character instead of the whole string.
2. Hamiltonian circuit is used as key to secure the data, hence decryption is practically incomprehensible unless the hamiltonian circuit and the encoding plan is known.

I think what's interesting is that in the above algorithm, I suggested them to provide a non-linear transformation to be able to achieve a higher security level. Because with the linear function, people can use brute force methods and eventually they will find out the plaintext. But this algorithm is also using a linear function but it makes it really hard to decrypt it using a brute force method.

But everything has pros and cons. It does achieve a high security level. But as a result, the complexity is really high.

5 Conclusion and future improvement

In this report, we talked bout 3 different kinds of encryption method using matrix.

Generally speaking, all three methods that we discussed have high computation overhead. This is because we need to recursively perform matrix multiplication. Graph theory in crptography has huge computation overhead since it encrypts the message character by character. And (optimized) hill cipher has the lowest computation overhead but it has low security level since it is easy for and adversaries to decrypt the matrix using a brute force method.

In order to provide an encryption method with strong security and low computation overhead, I think I will go for the second method and I will trying to add a non-linear transformation to provide stronger security. I didn't go fot the first method because it is way too simple, hence the security is still weak even though we provide a non-linear transformation. The third method has huge computation overhead. If we can encrypt the whole plaintext only once instead of character by character, hopefully we can have a huge improvement. But I don't think that can happen because we cannot concatenate the binary value of different numbers together.

References

[con22] Wikipedia contributors. Hill cipher, 2022.

[NMTG19] Nurhayati, Abdul Meizar, Frinto Tambunan, and Erwin Ginting. Optimizing the complexity of time in the process of multiplying matrices in the hill cipher algorithm using the strassen algorithm. In *2019 7th International Conference on Cyber and IT Service Management (CITSM)*, volume 7, pages 1–4, 2019.

[San14] Yeray Cachon Santana. Orthogonal matrix in cryptography. *CoRR*, abs/1401.5787, 2014.