Cryptography

Cheng-Jing Kuo
E-mail: jimkuo@aa.nctu.edu.tw
Graduate Institute of Communication Engineering
National Taiwan University, Taipei, Taiwan, ROC

Abstract

Not until several decades ago cryptology became science. For many centuries, it was a kind of secret technology practiced in black chamber by some special parties or governments. Secret key cryptography and public key cryptography are two main ideas to do encryptions. For the secret key system, we introduce two basic techniques. And we also introduce several important modes to show how we implement block ciphers. We also give an LFSR example for showing how the stream cipher works. We make a figure to compare the advantages and the drawbacks between the block cipher modes, and hope this comparison would help people who first contact this field to get some useful background sense quickly. In the middle of this article, we talk a little about the cryptanalysis, considering things on the attacker side. In there, we list some typical main ways that we can choose to do cryptanalysis. Then we mention some useful encryption examples for image and video encryption. In the end, we post the simulation result from three typical encryptions.

1 Introduction

In this section we are going to introduce some important nouns and basic sense in this field we are discussing.

Cryptography is the study of mathematical techniques related to information security aspects such as confidentiality, data integrity, entity authentication, and data authentication.

Cryptology is the study of cryptography and cryptanalysis.

Cipher is the way to encrypt data.

Plaintext is the original data before being encrypted and the data of the encryption output is called ciphertext or cryptogram. The methods which used to encrypt plaintext is called ciphers.

Figure.1 Plaintext and Ciphertext

Cryptanalysis is the study of methods of breaking ciphers. And the way to attack the cipher can be simply divided into two parts described below.

• exhaustive attack: Also called Brute-force attack which is an attack that would try all possible keys until the attacker hits the right key.

For example: encrypting by a 3-digits-number key, the exhaustive attack for the encryption is trying from 000 to 999 until the right key hit.

• statistics attack: Comparing to the exhaustive attack, statistics attack is a kind of systematical attack after observing input and output or some other effects.

For example: power consuming.

We will discuss cryptanalysis later.

(2)Family of Alice and Bob：

When cryptologists talk about encryptions, there are some roles involved inside such as message sender, receiver or attackers. There is a simple way to distinct these roles by naming the roles. Starting with the alphabet are Alice and Bob, two parties wanting to communicate in a secure manner. When more people are in the communication group, Carol and Dave will be used. Eve is a passive attacker who can get the information from Alice and Bob. Mallory is another attacker who can get the information and even modify the data between Alice and Bob. Trent is a person who is trusted by all involved parties. Walter is a man who would protect Alice and Bob with some aspect. This is the basic sense about the communication family. For more information and detail about this, we can go checking out the reference [1] and [3].

(3) Some basic sense of Data Encryption：

There are three main purposes in cryptography listed as following: First, creating confidentiality. Second, giving authentication which is used to recognize if the message sender is the legal one or not. And the third is integrity.

Modern cryptography relies on Kerckhoff Principle: we should always assume that all details about the cipher are known to the enemy, the exact algorithm and all its inner workings, except one small piece of data called key. We will mention it after in the cryptanalysis section.

2 Secret Key & Public Key Cryptography

In this section, we talk about the main idea that how data encryption process processes. Then we discuss the difference between two cipher ways that are symmetric cipher (secret key cryptography) and the asymmetric cipher (public key cryptography).

Generally speaking, the two cipher ways above are the two key systems we use to encrypt in cryptography field.

Moreover, we can even implement the symmetric cipher with two kinds of cipher, that is the block cipher and the stream cipher.

(1) Cipher classes for Data encryption：

Data encryption main process:

Figure.2 Plaintext and Ciphertext

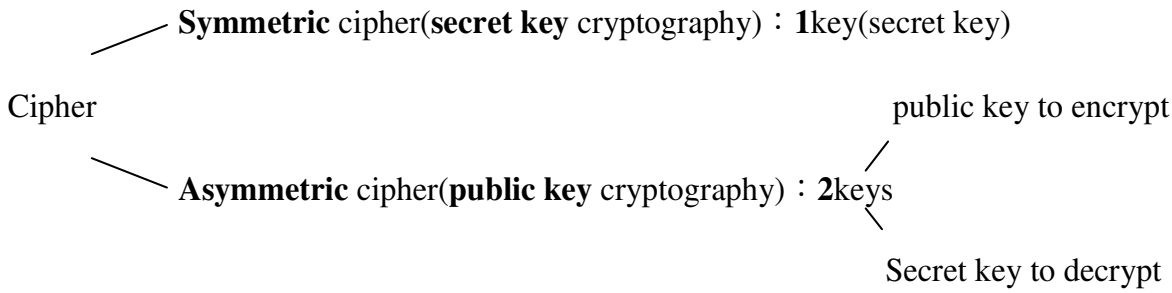

Figure.3 Symmetric cipher and Asymmetric cipher

In the secret key system, we use only one key to encrypt and decrypt. The

transmitters and the receivers have to own the same secret key. The concept of secret key is simple and fast. But it is hard to keep the key safety because as the involved parties number increase, the security of the secret key becomes unsafe. In order to solve this problem, someone invented the public key system or so called asymmetric cipher. There are two keys used in the cipher algorithm. The man owning the public key can encrypt the data, and only the man owning the secret key can decrypt the encrypted data. Asymmetric cipher is easy for key management, but the drawback is the computing speed is rather slow and complicated.

Table.1 advantage and disadvantage of symmetric cipher and asymmetric cipher

	advantage	**Disadvantage**
Symmetric cipher	fast and simple	key management is not easy
Asymmetric cipher	key management is easy	slow and complicated

3 Transposition Ciphers and Substitution Ciphers

From the encryption algorithm point of view, there are two main techniques we used to implement in the secret key cryptography (symmetric cipher) system: Substitution cipher and Transposition cipher.

Substitution ciphers replace bits, characters, or blocks of characters with substitution. Transposition ciphers rearrange bits or characters in the data. We now describe some details about the two kinds of cipher and simply introduce some examples that we use very often in the two kinds of cipher.

SUBSTITUTION TECHNIQUES

Substitution technique is one that the letters in the plaintext will be replaced by other letters or by numbers or symbols.

[Caesar Cipher]

The earliest use of substitution cipher is also the simplest one that is proposed by Julius Caesar, called Caesar Cipher. The Caesar Cipher works with replacing each letter with the letter standing three places further down of the alphabet order. For example:

plaintext: a b c d e f g h w x y z

ciphertext: e f g h i j k l z a b c

So if the plaintext is “meet me after the party”. The ciphertext would be “phhw ph diwhu wkh sduwb”.

plaintext: meet me after the party
ciphertext: phhw ph diwhu wkh sduwb

If we assign each letter a number from 0 to 25(from A to Z). Take the Ciphertext as C, Encryption as E, and plaintext as P. Then we can describe the Caesar Cipher as below

$$C=E(p)=(p+3)\mathrm{mod}(26) \quad (1)$$

A shift could be any amount, so the general Caesar algorithm is

$$C=E(p)=(p+k)\mathrm{mod}(26) \quad (2)$$

where k takes on a value in the range from 1 to 25. And the decryption algorithm is simply $p=D(C)=(C-k)\mathrm{mod}(26)$ (3)

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis will be easily performed. Just try all the 25 for the possible value of k.

In this example, there are three reasons for us to use the brute-force cryptanalysis. First is that the encryption and the decryption algorithms are known. Second is that there are only 25 keys to try. Third is that the language of the plaintext is known and easily recognizable.

For general cases, we always assume that the first condition is held, that is the algorithms of encryption and decryption are always known by the enemy who want to break the cipher. What really makes the brute-force attack impractical is that most of the algorithms use a large number of keys, that is, the second condition. For example, the triple DES algorithm uses a 168-bit key which makes people who choose to use the brute-force attacking way wasting resources or time. And the third condition is also important. If the language of the plaintext is unknown, we do not have any idea to recognize that if the key we try is right even in the trial that is right.

[Polyalphabetic cipher]

Simple substitution ciphers like Caesar cipher use a single mapping from plaintext to ciphertext letters, that is the same plaintext will have the same ciphertext. This characteristic is always not good in cryptography from the security point of view. Polyalphabetic cipher solves this problem by using multiple substitutions.

Image a cipher disk with two circles (outer and inner circle) and they are movable between each other.

Every time we randomly turn around the inner circle, we will get a response pair from each alphabet. Then we record where the &(or any sign different from alphabets and numbers) sign stand. That is the simple way to produce a substitution cipher which works and avoid the single mapping from plaintext to ciphertext problem. **(figure in Denning p.73)**

TRANSPOSITION TECHNIQUES

Transposition technique is achieved by performing some kind of permutation on the plaintext letters. It is very simple to realize this kind of cipher. We can do it by the example. If the plaintext is “meet me after the party”, we can rearrange it by this way:

m e m a t r h p r y

e t e f e t e a t

So we get the plaintext and the ciphertext like this:

plaintext: meet me after the party

ciphertext: mematrhpryetefeteat

[Columnar transposition]

Another simple transposition cipher is called Columnar transposition. If the plaintext is “data encryption”, we will compose the sentence into a 3*5 matrix. For example:

key: 4 1 2 3 5

plaintext : d a t a

e n c r y

p t i o n

ciphertext: anttciarodep yn

Of course, the transposition cipher can be made more secure by performing more than one stage of transposition. For example, doing the Columnar transposition 2 or 3 times and it will efficiently to increase the security of this cipher.

4 Block Cipher and Stream Cipher

In detail, we can even more separate symmetric cipher to two kinds of cipher as block cipher and stream cipher by the encryption basic sense. In this report we pay more attention to the block cipher, but we also give some stream cipher examples.

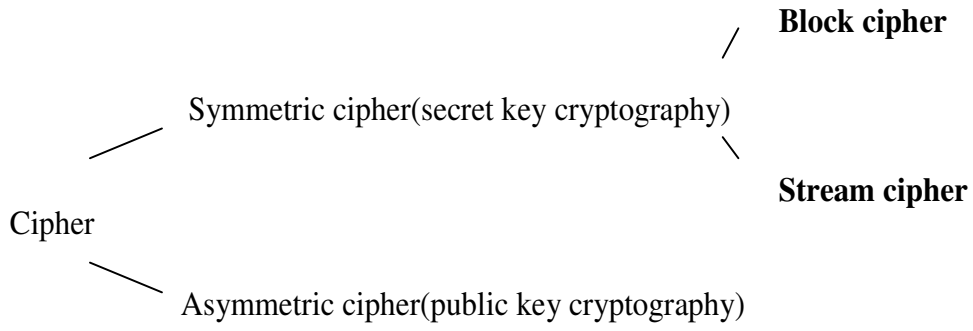

Figure.4 Block cipher and Stream cipher

The most different part between the block cipher and the stream cipher is that the block cipher encrypts the fixed size of the input data. On the other hand, stream cipher

encrypts the bitstream with the unit of bit or byte.

• **Block cipher**：

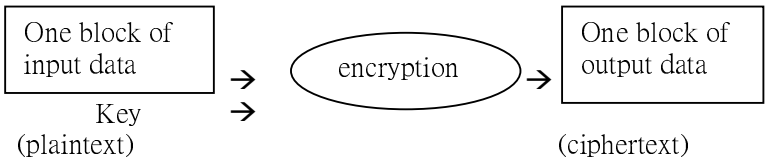

Figure.5 Block cipher scheme

Let M be a plaintext message. A block cipher breaks M into successive blocks $M_1, M_2, \ldots\ldots$ and encrypt each M_k with the same key K; that is,

$$E_k(M) = E(M_1)E(M_2).....\quad (4)$$

Typical size of block cipher block size is 64bits, 128bits or larger. Older cipher usually had the smaller size. Considering of the security, the larger the block size has, the safer the data is. Because each bits in the original data influences the every single output bit. And with aspect of processing speed, it is the same that we hope that the block size much larger. One of the advantages of the block cipher is the fast speed.

The drawback of the block cipher is that we must fit the block size, or we cannot do block cipher encryption. Sometimes we have to add additional redundant to fit the block size to do encryption. And this is kind of wasting resource.

• **Stream cipher**：

Stream cipher is different from block cipher that stream cipher break message M into successive characters or bits $m_1, m_2, \ldots\ldots$ and encrypt each m_k with the ith element k_i of a key stream K= $k_1 k_2 \ldots\ldots$; that is,

$$E_k(M) = E_{k_1}(m_1)E_{k_2}(m_2).....\quad (5)$$

The stream cipher produces key stream by using a key instead of dealing with block data. The key stream is often used to do XOR with plaintext and the results could be used to do encryption. We describe the XOR algorithm as followed.

(2)XOR(exclusive or)

XOR-operation $\oplus$ is often used in the cryptology.

Table.2 XOR-operation

Input 1	Input 2	XOR output
0	0	0
0	1	1
1	0	1
1	1	0

We explain how to do a simple data encryption using XOR by the followed example：

T: plaintext bitstream. S: some secret bitstream. We define C as the result of T⊕S, C=T⊕S. And now C is the cipher text. Then we can decrypt the encrypted data by the following steps:

C⊕S=(T⊕S) ⊕S=T⊕ (S⊕S)=T⊕0=T，So T=C⊕S。

We can decrypt the encrypted data by doing C⊕S.

5 Operation Modes for Block ciphers

Here we simply introduce some modes to implement block ciphers. These different modes we call them ”Operation Modes”. We choose one of them to implement the block cipher by considering the different kind of outstanding threatens.

ECB(electronic codebook mode)：

The simplest sense of Block cipher is ECB mode. In ECB mode, each encryption and decryption of the data blocks are independent from one another. It means that the speed of ECB mode is very fast because the parallel inputs and parallel outputs could be used. And the transmission errors will be confined inside the single block, and will not influence on the other blocks. The drawback of ECB mode is that the same plaintext input will have the same ciphertext output. It would be an advantage that the attackers could take on.

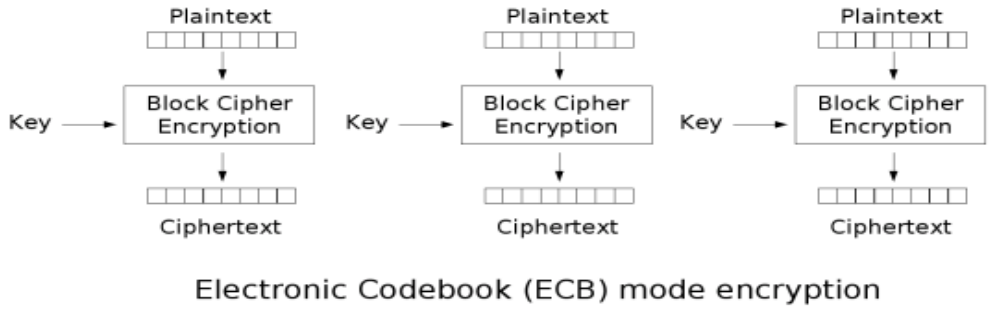

Figure.6 ECB mode encryption

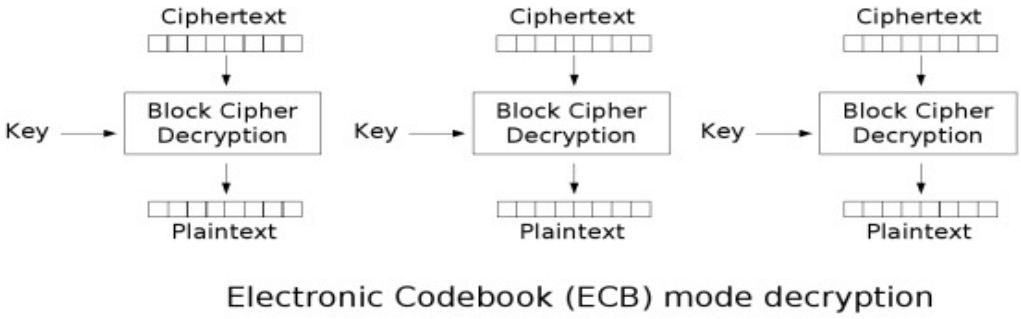

Figure.7 ECB mode decryption

CBC(cipher block chaining mode)：

CBC mode efficiently solves the security problem of ECB. The encryption of CBC is to do XOR between the current plaintext and the former plaintext, then deal the result from the above with the key. And the output is the current cipher. Decryption is quite simple that we could use the specification of the XOR. We could realize the detail by checking out Fig.8 and Fig.9. (Notice that there is a IV, initialization vector in the first step where there is no former ciphertext.)The disadvantage of CBC is that the processing speed in CBC is slower than ECB because the parrell inputs cannot be used here.

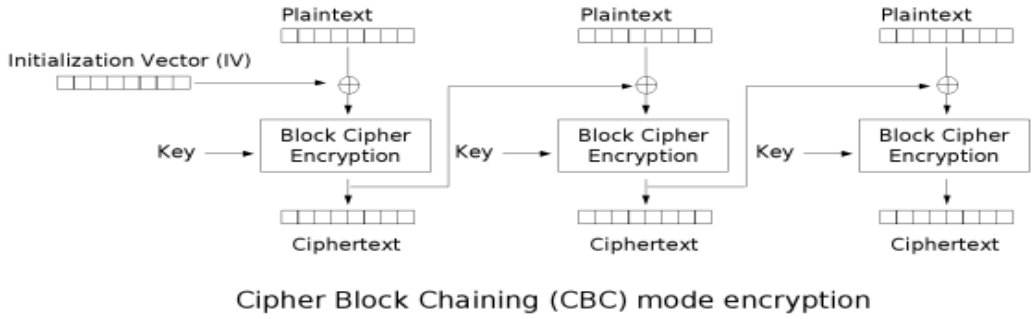

Figure.8 CBC mode encryption

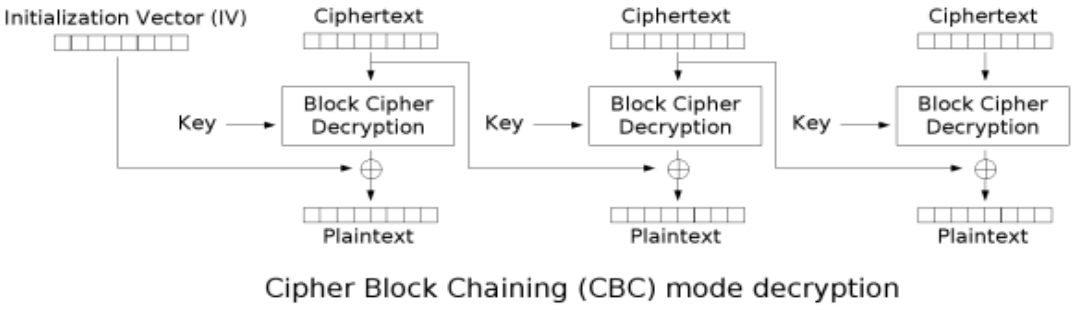

Figure.9 CBC mode decryption

CFB(cipher feedback mode)：

The most serious problem (drawback)is that it can only encrypt the data fitting the data size. CFB could solve this problem. CFB can deal with any data that even smaller than the block size. On the other hand, we can image this is a way transferring block cipher to stream cipher.

Fig.10 is an example for 8-bit CFB. At the beginning, the former ciphertext(or IV) is put into a shift register(we assume that the register shifts from right to left.) and the stuff inside the register would be encrypted with key. In general, the encryption output of the left n bits is exactly the ciphertext(now it is 8 bits).And the decryption is as same as the former modes that using the XOR.

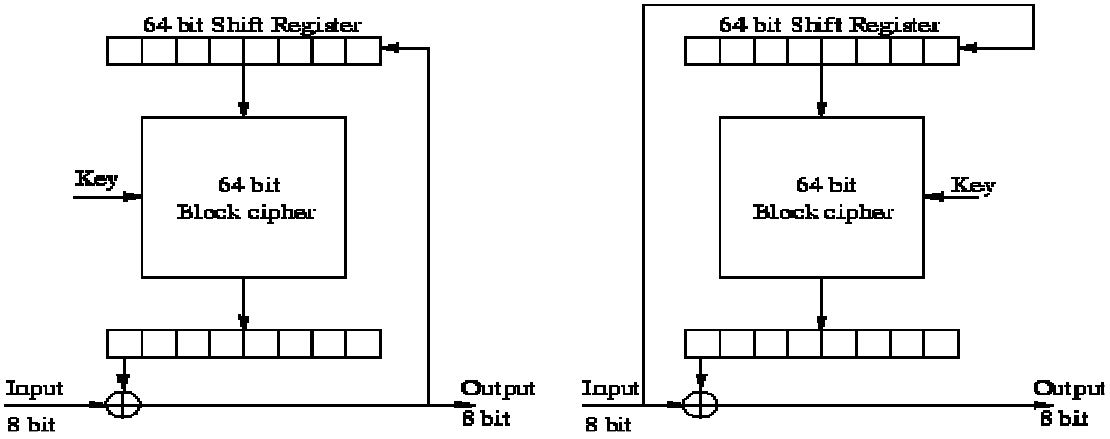

Figure.10 CFB encryption and CFB decryption

OFB(output feedback mode)：

OFB is similar to CFB that both of the two modes could transfer block cipher into stream cipher. The most difference between them is that OFB put the output of the encryption into to register directly. So OFB is a little simpler than CFB. check out the figure：

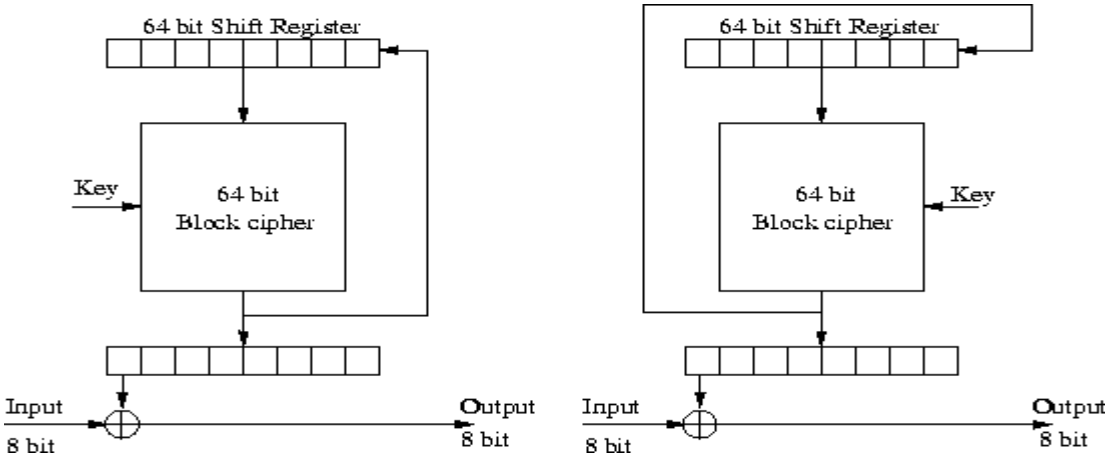

Figure.11 OFB encryption and decryption

CTR(counter mode or SIC, segment integer counter)：

The concept of CTR mode is also familiar to OFB. The difference between them is that there is no register inside the CTR system. Instead of the register, the CTR mode uses the counter to do encryption. The counter would be added by 1 every time after encryption. The biggest advantage of CTR is that the parallel inputs can be used which means the processing time of CTR mode is rather fast. And at the same time, there is no such security problem happened in ECB mode in CTR mode.

CTR is a popular mode that used very often nowadays. There is so many different type of such mode just like CTR. CTR is also suitable in the muti-processor machine. We just simply described the basic sense of CTR mode and put the simplest algorithm figure beyond.(The nonce here is the meaning as the IV above.)

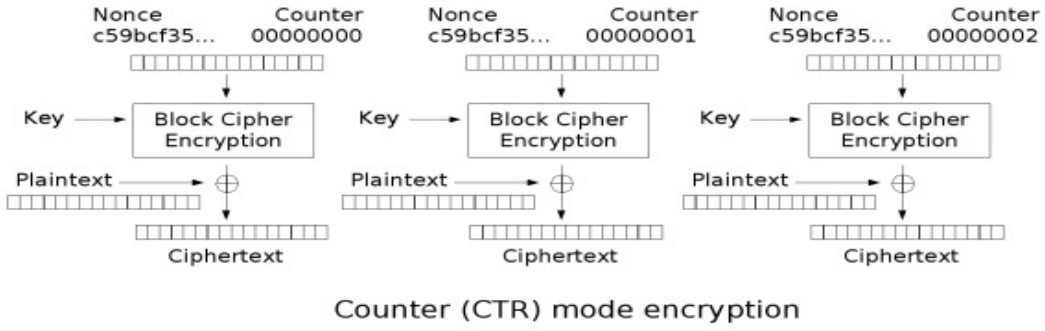

Figure.12 CTR mode encryption

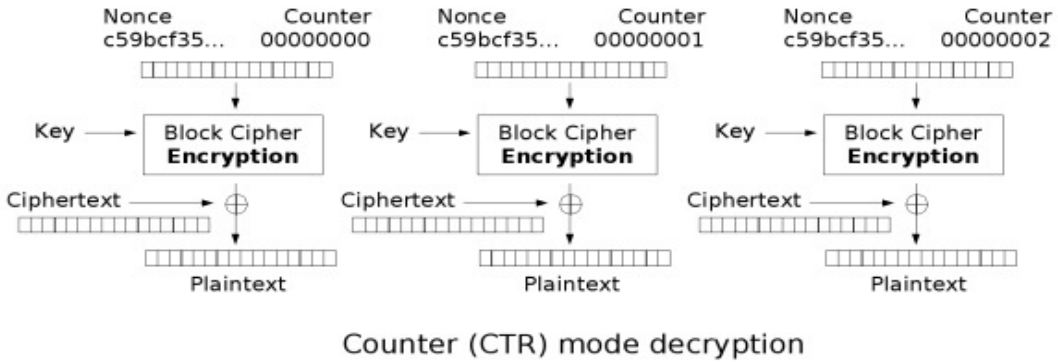

Figure.13 CTR mode decryption

Table.3 advantage and drawback of the modes above

	Advantage	**Drawback**
ECB	Parallel en/decryption, Simple and fast	Insecure, Handling constant data size
CBC	Secure	Series en/decryption, Slow, Handling constant data size
CFB	Secure, Handling data with any size	Series en/decryption, Slow
OFB	Secure, handling data with any size, Simpler and faster than CFB	Series en/decryption, Slow
CTR	Parallel en/decryption, Handling data with any size, Secure	

6 stream cipher example：LFSR and OTP

We mentioned the sense of stream cipher above. We now take LFSR for example. LFSR (linear feedback shift register) is so popular because the stream cipher here is easy to be implemented from the hardware point of the view.

There are two main parts of LFSR, shift register and feedback function.

1: Shift register

Shift register has two main jobs to deal with. The first is to deal with parallel or series data, and the second is to delay a serial bitstream. There are two ways to use the register. The first one is to input all the value into the register in one time (so called parallel input), then shift the output one by one (so called serial).The second one is serial input and parallel output. And the concept of delaying is rather simpler as it is the basic function of the shift register.

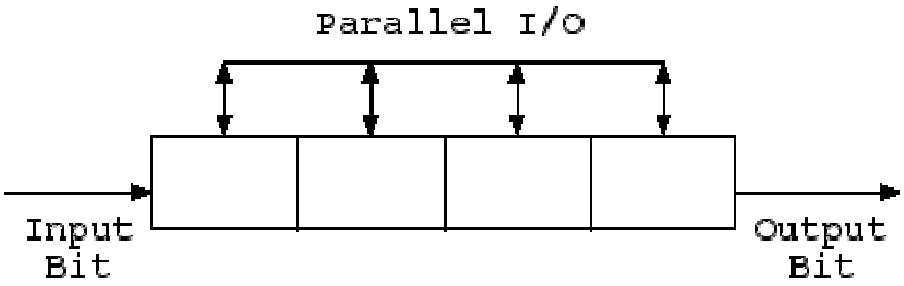

Figure.14

2: feedback function

The computation in the LFSR is so called the XOR, odd parity, or sum modulo 2.No matter what it is called, the concept is very simple：(1)add the select bit value(2)if the sum is odd, the output will be 1,or output 0 and vise versa.

The place we choose to do “add” is called the tap. The sequence of select bit tap is tap sequence. And the tap sequence would be feedbacked into the input of the register.

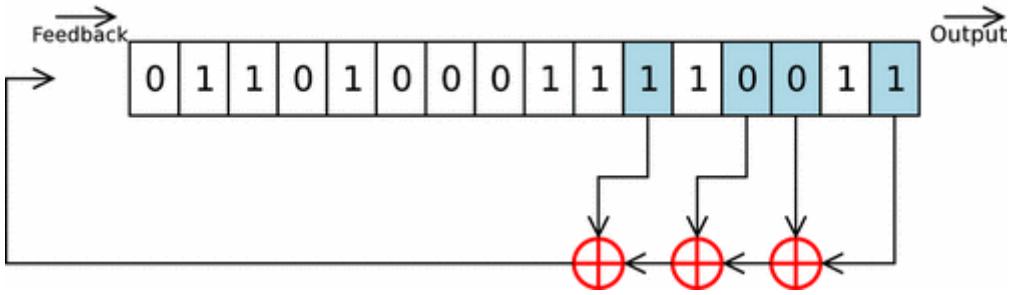

Figure.15

OTP (one time pad)is another example for stream cipher. The difference between OTP and the other stream ciphers is that the key stream produced by OTP will not be repeated. The specification of unrepeated key stream guanrantees the security, but also brings new problem, key distribution and randomness which means that we cannot use any software to produce the key stream generator because the software cannot modify the real random distribution. We have to use things such as atomic decay or radiation, etc to get the real random distribution.

7 Hybrid Algorithms, PGP

The most serious drawback of Public key algorithm is the complicated computation and the slow speed, but all the other specifics of this algorithm is very useful, especially in the condition that the communicators do not know each others. As the reason, we wonder if there were some ways to combine all the advantage of secret key and public key algorithms.

A widely used example for the above is PGP. The original plaintext is encrypted by a temporary key and the symmetric cipher system. The temporary key is produced

as random. Since the mechanism is symmetric cipher, the key should be transmitted to every parties involved in the communication system. We need to notice that the key would be encrypted by public key mechanism before it is transmitted. Therefore, if there are n receivers, there would be n versions symmetric key hidden in it.

This kind of cipher algorithm that combining both advantage of secret key and public key is so called Hybrid algorithm.

8 Cryptanalysis

Cryptanalysis is the methods to attack cryptographic protection. There are several ways to achieve this goal. A cipher is breakable if it is possible to determine the plaintext or key from the ciphertext, or to determine the key from the plaintext-ciphertext pair. There is a kind of attack called "Brute-force attack" which means that the attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried before succeeding. Besides Brute-force attack, there are three typical attacking situations we describe as followed.

1 ciphertext-only attack: the attackers only know the information of ciphertext and the detail of how encryption works, that is they know the encryption algorithm. (the Kerckhoff principle).

2 known-plaintext attack: the attackers know the encryption algorithm and the ciphertext just as cipher-only attackers know. And additional, they know one or more ciphertext – plaintext pairs generated by the secret key.

3 chosen-plaintext attack: the attackers not only know the encryption algorithm but also have the ability to modify the input plaintext and observe the corresponding output ciphertext.

Among the three situations, no doubts that the chosen-plaintext attack is the most threaten one in the point of view of the original data protectors.

A cipher is called "**unconditional secure**" if no matter how much the ciphertext is intercepted, there is not enough information to determine the corresponding plaintext uniquely. By the way, we have to realize that all ciphers are breakable if given unlimited resources.

So generally speaking, the "**computationally secure**" is sometimes more meaningful, which means if it can be broken by systematic analysis with available limited resources.

Computationally secure is established with the two criteria meet at the same time:

1. the cost of breaking the cipher exceeds the value of the encrypted information.
2. the time required to break the cipher exceeds the useful lifetime of the information.

Table.4 unconditional secure and computational secure

Unconditional secure	cannot be achieved in practical.
Computationally secure	criteria: 1. the cost of breaking the cipher exceeds the value of the encrypted information. 2. the time required to break the cipher exceeds the useful lifetime of the information.

Except the attack ways above, there are some attackers who do not consider pure cryptanalysis but mention about other effects .They might choose different inputs to observe the power consumption or the consuming time. They would guess the secret key by these kinds of way.

9 Assessment for image and video encryption

We have some critical properties for image and video encryption that we should evaluate such as “Time demand”, “Security”, “Bitstream compliance”, “Compressed domain processing”, “Compression performance affected”.

Table.5 important properties of image and video encryption

Time demand	Two main parts	1. Time(E): the time required for the actual encryption. 2. Time(P): the time required for the system to decide which parts of the data is going to be encrypted.	0, low, medium, high
Security	Two main aspects	1. The security of the cipher in use itself.	low, medium, high
Bitstream compliance	An image or video encryption scheme is said to be bitsrtream is compliant if the resulting bitstream is compliant to the bitstream definition of the compression system in use.		yes, no
Bitstream processing	Whether the encryption is applied directly to the bitstream or the bitstream needs to be encoded before encryption.		yes, no
Compression	A lot of image and video encryption schemes		yes,

performance affected	increase the file size as compared to applying compression without encryption.	moderately no

10 Algorithms for DCT-based Tech Image Encryption

(1) Zig-Zag Permutation Algorithm

Zig-Zag permutation algorithm is one of the common used compression oriented scheme image encryption and is the first MPEG encryption. Its main idea simple that is to substitute the fixed zig-zag quantized DCT coefficient scan pattern by a random permutation. We have to mention that the JPEG and MPEG standard orders the coefficients with respect to increasing frequency and decreasing magnitude. Therefore, long runs zeros occur in the high frequency areas of block. We could expect to lost compression performance.

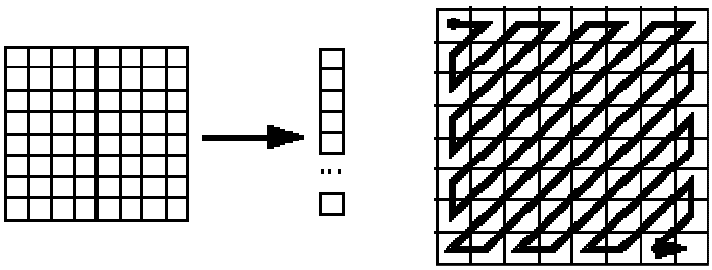

Figure.16

Tab.6 the properties of Zig-Zag algorithm

Time(E)	Time(P)	Security	BS compl.	BS proc.	CP affect
Low	0	Low	Yes	No	Yes

(2) Frequency-band Coefficient Shuffling

In order to improve the security and the limit the drop in compression efficiency as seen with zig-zag permutation, some people propose not to permute the coefficients within a single 8*8 pixels block but to group the entire set of similar frequency location coefficients together and perform permutation on that. It improves more security than the pure zig-zag permutation because more blocks are involved and additional key material may be employed to define which blocks should be selected to encrypt.

Tab.7 the properties of Freq-band shuffling

Time(E)	Time(P)	Security	BS compl.	BS proc.	CP affect
Low	Low	Low	Yes	No	Moderately

(3) Coefficient Sign Bit Encryption

This encryption encrypts the sign bit of each DCT coefficient only. The main idea of this encryption is that by changing the sign bit of all DCT coefficient, the output of the encryption remains high entropy. Moreover, we could even encrypt the DCT coefficients with changing the sign bits and in additional to do an extra zig –zag permutation.

Tab.8 the properties of Coefficient sign bit encryption

Time(E)	Time(P)	Security	BS compl.	BS proc.	CP affect
Medium	Medium	Low	Yes	Yes	No

(4) Secret Fourier Transform Domain

All of the above we talk about the encryptions that deal with the transform domain coefficients. So if the transform domain is not known by the attackers, it will be very hard for the attackers to attack the encryption. One of the examples is that the input plane, the encryption plane, and the output plane are the fractional Fourier transform to one another. General speaking, the most serious problem of this kind of encryption is that it might take a lot of time for the system to compute throughout all the transforms.

Tab.9 the properties of secret FT domain

Time(E)	Time(P)	Security	BS compl.	BS proc.	CP affect
High	High	High	No	No	Yes

11 Algorithms for DCT-based Tech Video Encryption

(1) Encryption of I-frames

Some people propose to encrypt I-frames only. Since P-frames and B-frames are restructed from the predictions of I-frames, it is reasonable to assume that if I-frames are encrypted, P and B-frames are expected to be protected as well. But there are several problems we should mention. First, the I-frames is about 25-50% in the whole frames which means this encryption approach does not reduce the computational complexity enough. Second, the motion in the video remains visible, especially when replacing the encrypted I-frames by uniform frames. We could solve the first problem by increasing the numbers of I-frames, but that makes the second problem be more serious. It means that we would have to trade off between security and the compression

performance.

Tab.10 the properties of I-frame encryption

Time(E)	Time(P)	Security	BS compl.	BS pronc.	CP affect
Medium	Medium	Low	Yes	Yes	No

Simulation result of image encryption

We simulate three simple sense of image encryption by using the block permutation, zig-zag permutation and dealing with the coefficients using mod computation. All of the three algorithms do not transform the original image coefficients into another domain. We just make a trial on these three encryption sense and give examples above, so it is not the point that the domain is transformed or not. We show the encryption and decryption results below and discuss the performance after all.

- (Zig-Zag permutation encryption)

Figure.17 Original image: lena.jpg

Figure.18 Image after zig-zag permutation and the decryption image

- (Doing Mod computation on coefficients encryption)

Figure.19 Image after coefficient mod dealing and the decryption

- (Block permutation encryption)

Figure.20 Image after block permutation and the decryption

Advantage and Disadvantage

The block permutation encryption is the simplest idea that encrypts the image. But the security is a problem. We can conceal this problem by changing the permutation block size and increase the permutation numbers. The idea of mod way encryption encrypts the image by changing the coefficients of the image matrix in a mod computing formula (We can tune the formula as our will of course.). The problem of it is that the choice of coefficient in the formula is critical. In our program the coefficient of the mod formula cannot be chose too big, or the image after encryption will be simply seen as an image composed by several small size of the original image. The advantage of the mod way encryption is that the computational complexity is very low, that makes the encryption speed very fast. The zig-zag

encryption seems the best way to encrypt image among the three algorithms we discuss. And the computational complexity is in the acceptable range that will not yet be a problem. Moreover, we can combine the three algorithms together to encrypt images, and that would ensure the security well. The table below lists the advantages and the drawbacks of the three simulations we discuss.

Table.11 advantage and disadvantage of the simulate examples

	advantage	**disadvantage**
Block permutation	Simple to realize	Security problem
Mod coefficient encrypt	Low complexity	Security problem
Zig-zag permutation	Secure, Simple, Low complexity	

Conclusion

We have seen the basic background knowledge of cryptography and realized some image and video data encryption schemes and examples. Most parts of the report are emphasizing on the whole scheme of the data encryption process and the basic sense of cryptography and cryptanalysis. We should understand that there is always a "trade-off" to deal with the two aspects of security and the computational complexity when we encrypt the visual data. Also we have to know that the standards used nowadays such as MPEG IPMP or JPSEC are quite recent and are not going to be the final or the best standards. Indeed, there will always be some more powerful standards invented in the future. Consequently, almost all of the literatures suggest that we can pay more attention on the entertainment and telecommunication application which will possibly make more extensive use of encryption. And that would bring us more exciting ideas even from the commercial point of view.

References

[1] A. Uhl, A. Pommer, *Image and Video Encryption*, Springer, City, 2005

Chap3.Cryptography 基本架構，Cryptanalysis overview。

Chap4.DCT-based 架構下發展 image 與 video encryption 演算法的重要考慮元素。

此書薄，從 cryptography 到 data compression 再到 data encryption 都有包涵，初學者看較爲吃力，但因去除許多細節，已有概念者閱讀起來會很輕鬆，易心領神會。

[2]D. Elizabeth, R. Denning *Cryptography and Data Security*, Addison-Wesley, City, 1982

非常早期的書，此報告參考其 secret-key system 中兩個分類：transposition cipher

technique，substitution cipher technique。

[3] W. Stallings, *Cryptography and Network Security*, Prentice Hall, 2003

Chap2.Block cipher。

Chap3.Block cipher modes。

Chap9.Public-Key cryptosystems。

適合當作 Cryptography 的入門書，從基本概念介紹到 Public-Key Secret-Key 架構，還有 Cipher 的分類，Key Management，Message Authentication，最後兩個章節分別談到 Network Security 與 System Security，適合初學者閱讀，且作者 William Stallings 在 Cryptography 領域有好幾本相關的書籍。

[4] P. Wayner, *Disappearing Cryptography Information Hiding: Steganography & Watermarking,* Morgan Kaufmann, 2002

[5] D. Salomon, *Data Compression*, Springer, 2004

Chap 6-4. I-frame，P-frame，B-frame 的概念。

是一本完整的 data compression 參考書(The Complete Reference)，介紹各種 statistical coding 方法，Image compression，Wavelet Methods，Video compression，Audio compression 等重要的領域，對各種演算法也有詳細的圖文解釋。

[6] L.Tang, "Method for Encrypting and Decrypting MPEG Video data Efficiently", In *Proceedings of The ACM Mutimedia*, Publisher, City, 1996

有介紹到 Zig-Zag permutation algorithm

[7] W. Zeng, S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video", *IEEE Transactions on Mutimedia*, 2003

http://ieeexplore.ieee.org/iel5/6046/26936/01196741.pdf?tp=&isnumber=&arnumber=1196741

提到 Frequency-band Coefficient Shuffling 與 Coefficient Sign Bit Encryption。

[8] W. Zeng, S. Lei, "Efficient Frequency Domain Selective Scrambling for content access control", In Proceedings of The Seventh ACM International Mutimedia Conference, 1999

同上，提到 Frequency-band Coefficient Shuffling 與 Coefficient Sign Bit Encryption.

[9] A.J. Menezes, P.V. Oorschot, S.A. Vanston, *Handbook of Applied Cryptography*, CRC Press, 1996

講到部份關於 Crypyanalysis，Cryptography 的基本概念

[10]B.Schneier, *"Applied Cryptography: protocols, algorithms and source code in C"*, Wiley Publishers, 1996

簡單提及 The family of Alice and Bob 的概念

[11]A. Kerckhoff, *"La cryptographie militaire"*, Journal des sciences militaries, 1883

Kerckhoff Principle：The attackers always know the encryption algorithm detail.

[12]N. Drakos, A. Eng, 'Secure Telnet', available from http://www.pvv.ntnu.no/~asgaut/crypto/thesis/thesis.html

取其 Block cipher modes 的圖

[13]R. Blaschke, ‘Kryptographie -- Multimedia Security, MPEG’, available from http://www.rblasch.org/studies/crypto/index.html

取其 zig-zag permutation 的圖

[14] H. Cheng and X. Li. On the application of image decomposition to image compression and encryption. In P. Horster, editor, *Communications and Multimedia Security II, IFIP TC6/TC11 Second Joint Working Conference on Communications and Multimedia Security*, CMS ’96

參考 Data compression 的部分。

程式片段：(附上 3 種此報告內撰寫之 image en/decryption 方法)

1.Zig-Zag permutation encryption(fig.18)

Encryption：

```
function En=jimkuozigzag(T,a,b,c)
N=size(T);
%allpix=N(1)*N(2)
%newpix=sqrt(allpix);

%此 zigzag 函式用來將輸入的 image 利用 zigzag 掃描方式加密
% 其簡單概念如下:
% (1)輸入 image 'T'
% (2)用 zigzag 方式將 T 矩陣的值掃描成一個一維 array  'new'
% (3)將此一維 array 利用 reshape 語法,轉成與 T 相同維度大小的矩陣 En,即爲加密之新圖片
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 程式做法說明
% 基於 ZigZag 的掃描方式,每一條的掃描用變數 d 來計算
% new array 的 index 以 n 來計算,(1,1)的值輸入給 new(1),(1,2)輸給 new(2)..etc.
% 第一條掃描 d=1,zigzag 掃描路徑即爲(1,1)-->(1,2)
% 第二條掃描 d=2,zigzag 掃描路徑爲(1,2)-->(2,1)-->(3,1)
%            注意最後(3,1)此處,此時 d 仍然是 2,我先把(x,y)改變成下一條的起始點(3,1),稱
'掃描',但不作輸入動作
%            到 d=3 才做將(3,1)輸入 new 的動作.
%            所以,以 d=2 爲例,程式的動作就是:(1)先將 d=2 起始點的值輸入給 new
%                                        (2)先做輸入,再做掃描
%                                        (3)到左邊邊緣後(此判斷式即爲 while 迴圈
y<=1 成立,迴圈停止)
%                                        (4)輸入值給 new,再做座標改變(d=3 的起始座
標)
% 第三條掃描 d=3,爲(3,1)-->(2,2)-->(1,3)-->(1,4)
% 以此類推...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=1;y=1;n=1;                   % x,y 爲座標,n 爲 new 的 index,亦是現在掃描於第幾個
pixel 的數目
new(1)=T(1,1,1);               % 先將(1,1)輸入 new(1),並 y++此爲第一路徑輸入與掃描
y=y+1; n=n+1;                  % 開始迴圈以前已經將第一路徑輸入掃描完畢
```

```
                                % 所以,d=1 時,輸入動作爲 new(1)=T(1,1,1),而掃描動作爲 y=y+1
                                % 即從(1,1)-->(1,2)

for d=2:N(1)                    % 迴圈開始,由 d=2 開始
    if mod(d,2)==0              % 用 d 的單偶數來判斷 zigzag 的方向
        while y>1               % while 條件式來判斷是否已經接觸到最左邊
            new(n)=T(x,y,1);    % 現在 d 爲偶數,掃描方向爲右上到左下,所以每次先將座標值輸入 new,再做掃描:將 x++,y--,n++,
            x=x+1;y=y-1;
            n=n+1;
        end                     % 當 y<=1,即碰到最左邊,此時輸入 new,再做 x++的動作(例:d=2 時 new(3)=T(2,1,1),再 x++,即(x,y)變爲(3,1))
        new(n)=T(x,y,1);
        x=x+1; n=n+1;
    else                        % 此時 d 爲單數,表示 zigzag 方向由左下到右上,因此迴圈內在掃描碰到最上面(x>=1)以前,每次掃描都爲 x--,y++
        while x>1
            new(n)=T(x,y,1);    % 同樣若 d=3,先輸入 T(3,1,1)進入 new,再做 x--,y++
            x=x-1;y=y+1;
            n=n+1;
        end
        new(n)=T(x,y,1);        % 當 x<=1,即碰到最上邊,此時輸入 new,再做 y++的動作(例:d=3 時 new(3)=T(1,3,1),再 y++,即(x,y)變爲(1,4))
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;                   % 此爲剛好掃完一半的點,若爲N*N矩陣,此時在(N,1)位置,將 x--,y++

d1=N(1)+1; d2=2*N-1;            % 此開始與上面類似,僅因爲現在掃的是下三角的矩陣,所以 while 判斷邊緣有點改變而已
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            new(n)=T(x,y,1);
            x=x-1;y=y+1;
```

```
            n=n+1;
        end
         new(n)=T(x,y,1);
         x=x+1;n=n+1;
    else
        while x<temp1
            new(n)=T(x,y,1);
            x=x+1;y=y-1;
            n=n+1;
        end
        new(n)=T(x,y,1);
        y=y+1; n=n+1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%以下為 RGB 中的 GB 部分,方式同上,可直接跳至最後第 170
行的注解%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
x=1;y=1;n=1;
new2(1)=T(1,1,2);
y=y+1; n=n+1;
for d=2:N(1)
    if mod(d,2)==0
        while y>1
            new2(n)=T(x,y,2);
            x=x+1;y=y-1;
            n=n+1;
        end
        new2(n)=T(x,y,2);
        x=x+1; n=n+1;
    else
        while x>1
            new2(n)=T(x,y,2);
            x=x-1;y=y+1;
            n=n+1;
        end
        new2(n)=T(x,y,2);
```

```
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;

d1=N(1)+1; d2=2*N-1;
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            new2(n)=T(x,y,2);
            x=x-1;y=y+1;
            n=n+1;
        end
         new2(n)=T(x,y,2);
         x=x+1;n=n+1;
    else
        while x<temp1
            new2(n)=T(x,y,2);
            x=x+1;y=y-1;
            n=n+1;
        end
        new2(n)=T(x,y,2);
        y=y+1; n=n+1;
    end
end


x=1;y=1;n=1;
new3(1)=T(1,1,3);
y=y+1; n=n+1;
for d=2:N(1)
    if mod(d,2)==0
        while y>1
            new3(n)=T(x,y,3);
            x=x+1;y=y-1;
            n=n+1;
        end
```

```
        new3(n)=T(x,y,3);
        x=x+1; n=n+1;
    else
        while x>1
            new3(n)=T(x,y,3);
            x=x-1;y=y+1;
            n=n+1;
        end
        new3(n)=T(x,y,3);
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;

d1=N(1)+1; d2=2*N-1;
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            new3(n)=T(x,y,3);
            x=x-1;y=y+1;
            n=n+1;
        end
         new3(n)=T(x,y,3);
         x=x+1;n=n+1;
    else
        while x<temp1
            new3(n)=T(x,y,3);
            x=x+1;y=y-1;
            n=n+1;
        end
        new3(n)=T(x,y,3);
        y=y+1; n=n+1;
    end
end

%%利用 reshape 的方式,將一維的 new 改成與 T 同維的 matrix
%%再將 RGB 各 channel 指定給 D,最後 En 即爲輸出之加密圖像
```

```
C1=reshape(new,temp1,temp1);
C2=reshape(new2,temp1,temp1);
C3=reshape(new3,temp1,temp1);

D(:,:,1)=C1;
D(:,:,2)=C2;
D(:,:,3)=C3;

En=D(:,:,[a,b,c]);
```

Decryption：

```
function De=jimkuoZigZagDe(H,j,k,p)
%%此函式爲針對 ZigZag 加密的圖像做解密的動作
%先將輸入的 H 改成一維的 a
%再利用 jimkuozigzag 的各迴圈來做一樣的事情,只是在這邊是將 1 維的 a 指給 matrix T
%其餘與加密法都相同
N=size(H);
N1=N(1);
Dim=N1*N1;
a=reshape(H(:,:,1),1,Dim);
a2=reshape(H(:,:,2),1,Dim);
a3=reshape(H(:,:,3),1,Dim);

x=1;y=1;n=1;

T(1,1,1)=a(1);
y=y+1; n=n+1;
for d=2:N1
    if mod(d,2)==0
        while y>1
            T(x,y,1)=a(n);
            x=x+1;y=y-1;
            n=n+1;
        end
        T(x,y,1)=a(n);
        x=x+1; n=n+1;
    else
        while x>1
```

```
            T(x,y,1)=a(n);
            x=x-1;y=y+1;
            n=n+1;
        end
        T(x,y,1)=a(n);
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;

d1=N1+1; d2=2*N1-1;
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            T(x,y,1)=a(n);
            x=x-1;y=y+1;
            n=n+1;
        end
         T(x,y,1)=a(n);
         x=x+1;n=n+1;
    else
        while x<temp1
            T(x,y,1)=a(n);
            x=x+1;y=y-1;
            n=n+1;
        end
        T(x,y,1)=a(n);
        y=y+1; n=n+1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%下面爲其餘兩個 channel
x=1;y=1;n=1;

T(1,1,2)=a2(1);
y=y+1; n=n+1;
for d=2:N1
    if mod(d,2)==0
```

```
        while y>1
            T(x,y,2)=a2(n);
            x=x+1;y=y-1;
            n=n+1;
        end
        T(x,y,2)=a2(n);
        x=x+1; n=n+1;
    else
        while x>1
            T(x,y,2)=a2(n);
            x=x-1;y=y+1;
            n=n+1;
        end
        T(x,y,2)=a2(n);
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;

d1=N1+1; d2=2*N1-1;
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            T(x,y,2)=a2(n);
            x=x-1;y=y+1;
            n=n+1;
        end
         T(x,y,2)=a2(n);
         x=x+1;n=n+1;
    else
        while x<temp1
            T(x,y,2)=a2(n);
            x=x+1;y=y-1;
            n=n+1;
        end
        T(x,y,2)=a2(n);
        y=y+1; n=n+1;
```

```
    end
end

%%%%%%%%%%%%%
x=1;y=1;n=1;

T(1,1,3)=a3(1);
y=y+1; n=n+1;
for d=2:N1
    if mod(d,2)==0
        while y>1
            T(x,y,3)=a3(n);
            x=x+1;y=y-1;
            n=n+1;
        end
        T(x,y,3)=a3(n);
        x=x+1; n=n+1;
    else
        while x>1
            T(x,y,3)=a3(n);
            x=x-1;y=y+1;
            n=n+1;
        end
        T(x,y,3)=a3(n);
        y=y+1;n=n+1;
    end
end
x=x-1; y=y+1;

d1=N1+1; d2=2*N1-1;
temp1=N(1);
   for d=d1:d2
    if mod(d,2)==1
        while y<temp1
            T(x,y,3)=a3(n);
            x=x-1;y=y+1;
            n=n+1;
        end
```

```
            T(x,y,3)=a3(n);
            x=x+1;n=n+1;
        else
            while x<templ
                T(x,y,3)=a3(n);
                x=x+1;y=y-1;
                n=n+1;
            end
            T(x,y,3)=a3(n);
            y=y+1; n=n+1;
        end
    end

    De=T(:,:,[j,k,p]);
```

2.Doing Mod computation on coefficients encryption(fig.19)

Encryption：

```
function En=jimkuoModEnc(T,a,b,c)
pn=size(T);
 p1=mod(53*[0:pn(1)-1],pn(1))+1;
 p2=mod(173*[0:pn(2)-1],pn(2))+1;
 En=T(p1,p2,[a,b,c]);
% Function JimkuoModEnc is a function used to do picture encryption using
the MOD
% way to change the picture matrix element value.
% There are four input variables.
% You do not have to input the picture size bcz this function already do
% that for you.
% Variable T is the original picture file without encryption in the main
% function.
% And the a,b,c represent the R,G,B element,respectively.
% To use this function, you only have to input the pixel numbers of the
original
% picture,and see if you want to change the RGB element representing
% order with number 1 2 3.
% Then En is the picture results from MODway Encryption that we did by
the function.
```

Decryption：

```
function DeEn=jimkuoModDeEnc(En,a,b,c)
%p1=mod(53*[0:x-1],x)+1;
%p2=mod(47*[0:y-1],y)+1;
pn=size(En);
p1=mod(53*[0:pn(1)-1],pn(1))+1;
p2=mod(173*[0:pn(2)-1],pn(2))+1;
[a11,a12]=sort(p1);
[a21,a22]=sort(p2);
[temp,index]=sort([a,b,c]);
DeEn=En(a12,a22,index(:));
% This function is used to De-Encrypted pictures which be Encrypted by
the
% function JimkuoModEnc.m using the MOD way to change the picture matrix
% element value.
% To use this function, we also have to input the x-y pixel numbers of
```

```
the
% picture,and the RGB order we changed when encrypted the picture.
```

3. (Block permutation encryption) (fig.20)

X方向：

```
function En=jimkuoPermuencX(T)
pn=size(T);
x=floor(pn(1)/20);
y=floor(pn(2)/20);
Enc=T
i=1;
n=1;
%if  mod(x,2)=0
   while(n<=x/2)
     temp=Enc(i:i+10,:,:);
     Enc(i:i+10,:,:)=Enc(i+20:i+30,:,:);
     Enc(i+20:i+30,:,:)=temp;
     i=i+40;
     n=n+1;
   end

En=Enc;
```

Y方向：

```
function En=jimkuoPermuencY(T)
pn=size(T);
x=floor(pn(1)/20);
y=floor(pn(2)/20);
Enc=T
i=1;
n=1;
%%if mod(x,2)=0
   while(n<=y/2)
     temp=Enc(:,i:i+10,:);
     Enc(:,i:i+10,:)=Enc(:,i+20:i+30,:);
     Enc(:,i+20:i+30,:)=temp;
     i=i+40;
     n=n+1;
   end
 En=Enc;
```