Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP

Môn học: Mật mã học

Kỳ báo cáo: Buổi 01 (Session 01)

Tên chủ đề: Thuật toán mã hoá DES

Ngày báo cáo: 14/03/2023

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: **NT219.N21.ANTN**

STT	Họ và tên	MSSV	Email
1	Phạm Nguyễn Hải Anh	21502586	21520586@gm.uit.edu.vn
2	Nguyễn Hoàng Hải	1552	hochai12@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Câu hỏi 01	100%	
2	Câu hỏi 02	50%	
3	Câu hỏi 03	100%	
4	Câu hỏi 04	90%	
5	Câu hỏi 05	60%	

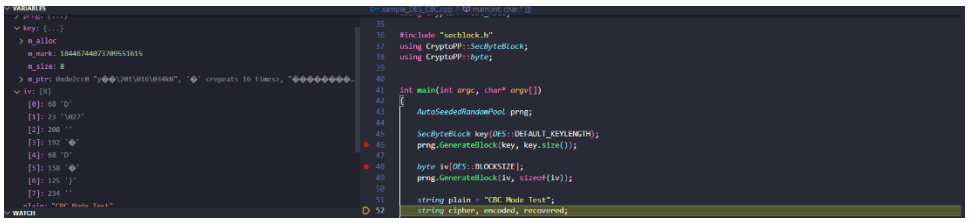
Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Exercise 1

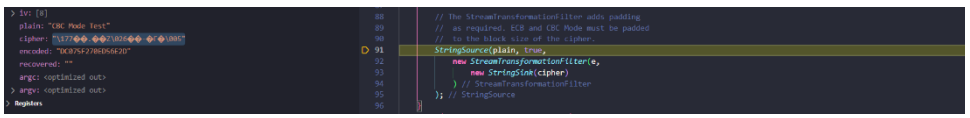
- generate 'key' and 'iv' variables by randomization using GenerateBlock() function, we get the result as string bytes, but the data type of cryptpp library is difficult to read.



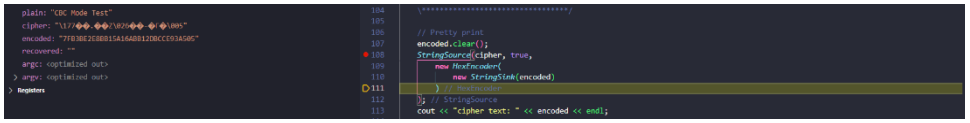
- then, to make it clear to the viewer, we convert the 'key' and 'iv' variables to a Hex type storage in 'encoded' variable and print it

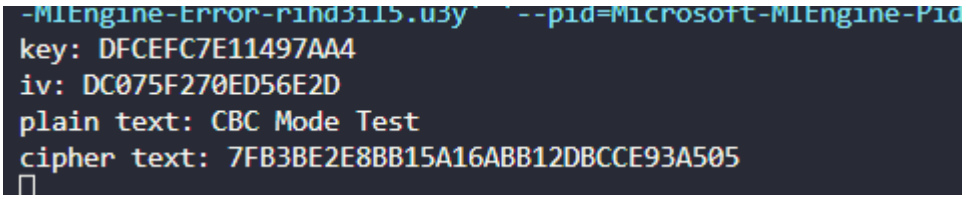
- then use CBC_Mode with 'key' and 'iv' , using SetKeyWithIV() function to set key and iv to 'e' variable

- encrypting 'plain' via StringSource(), StreamTransformationFilter() functions with 'e' parameter, then we get 'cipher' variable which is encrypted



- in order to be able to easily read the information of the cryptographic variable, we encode the 'cipher' into Hex and save it in the 'encoded' variable and then print 'encoded'




-MIEngine-Error-rihd3115.u3y --pid=Microsoft-MIEngine-Pid
key: DFCEFC7E11497AA4
iv: DC075F270ED56E2D
plain text: CBC Mode Test
cipher text: 7FB3BE2E8BB15A16ABB12DBCCE93A505

2. Exercise 2

Main activities of the DES decryption with CBC mode:

(I do not know how to Debugging with vscode, so instead I explain the logic of the decryiption.) Because each time I run code, different permutation occurs, so in this scenario, I suppose the receiver receives the key = 12345678_{ASCII} (this number won't have correct parity bits) and iv = $abcdabcd_{ASCII}$ and ciphertext.

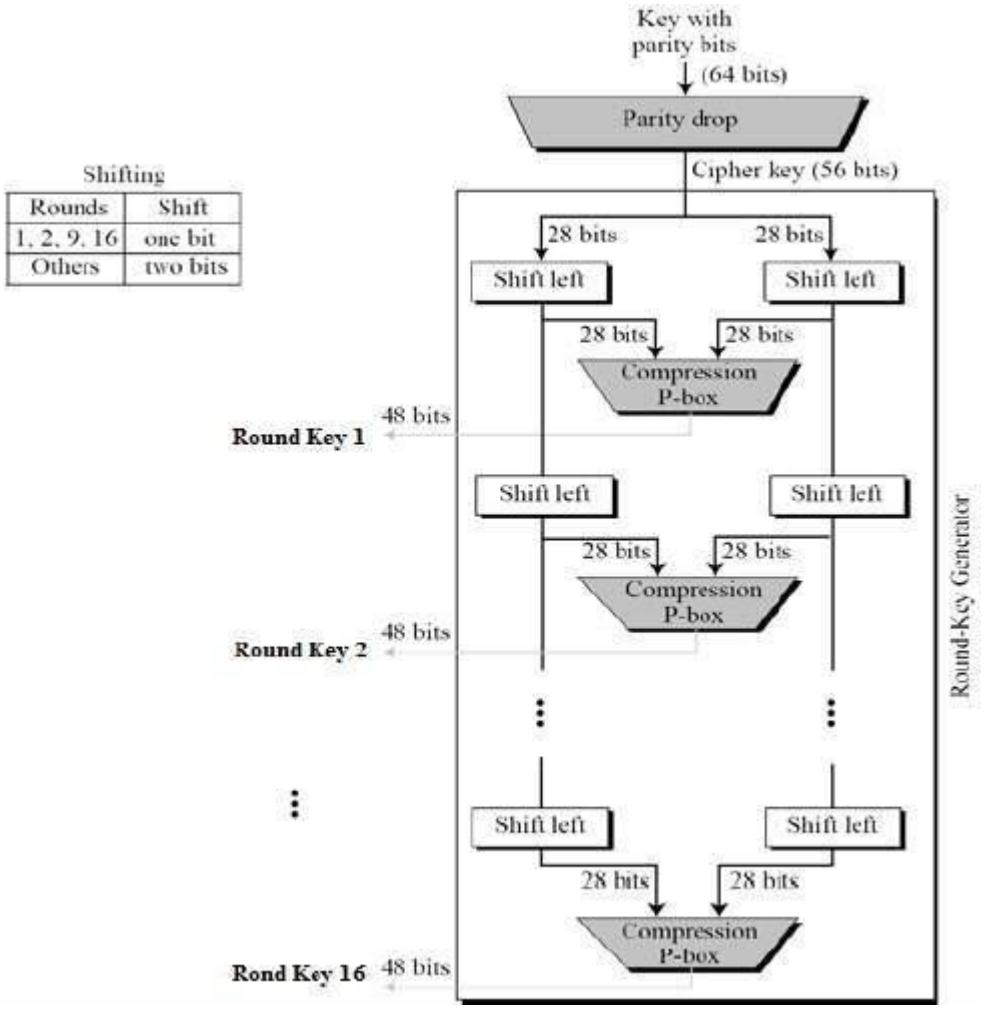
key = 00110001 00110010 00110011 00110100 00110101 00110110 00110111 00111000

iv = 01100001 01100010 01100011 01100100 01100001 01100010 01100011 01100100

ciphertext = $64\ 66\ 64\ 73\ 61\ 77\ 72\ 65\ 77\ 66\ 61_{hex}$ = 1100100011001100110010001110011011000010111011101110010011001010111011101100110011001100001

In the decryption, the ciphertext will be separated into 64-bit blocks, and 16 subkeys (RoundKey) will be generated as the same way in the encryption. And the 16^{th} subkey will be used for the first round of decryption. It is reverse order to the encryption, which uses the 1^{st} subkey for encrypting.

Key generating process:



Key generation process

At first, all 8x bits (8, 16, 24, 32, …, 64) will be dropped, or used for parity. Then PC-1: 64 – 8 = 56 bits left will be permutated, then divided into 2 halves (L & R), each with 28 bits:

(As {a, b, c} is a permutation of {a, b, c}, I suppose the key does not change after permutated)

Key = 0011000 0011001 0011001 0011010 | 0011010 0011011 0011011 0011100

L = 0011000 0011001 0011001 0011010

R = 0011010 0011011 0011011 0011100

Then rotate left for both L and R. Round 1 will shift left 1 bit:

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

shift left bits convention

L = 011000 0011001 0011001 0011010 0

R = 011010 0011011 0011011 0011100 0

Then PC-2 (like PC-1) but only 48 bits left from both L and R. They are considered as the first 48-bit subkey, however, used in the last decryption round.

Assume first subkey is

011000 0011001 0011001 0011010 0 011010 0011011 0011011

Continue with the second subkey with the same PC-1, PC-2 and same last L and R.

...

Assume the 16th subkey is

011010 0011011 0011011 0011100 0011000 0011001 0011001

Decryption process:

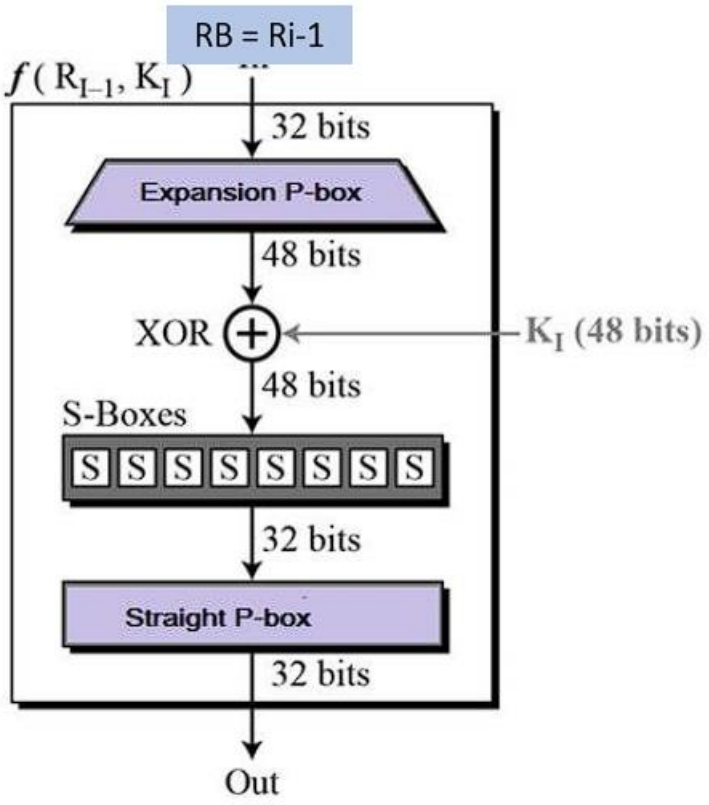
The first 64-bit block of ciphertext is:

1100100011001100110010001110011011000010111011101110010011001010

The Initial Permutation (IP) and Final Permutation (FP, which inverses the IP) is just used to support the hardware, so I assume that after IP (FP) the block does not change, for short. The 64-bit block will be divided into 2 32-bit blocks, LB & RB.

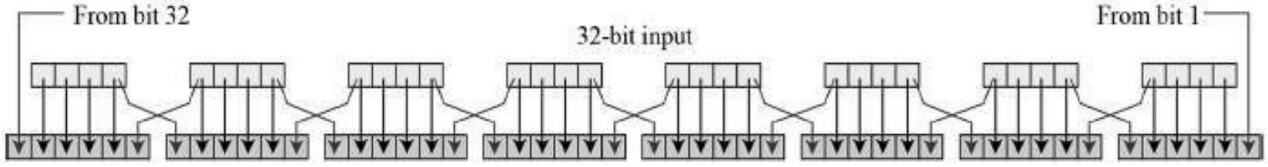
LB: 11001000110011001100100011100110

RB: 11000010111011101110010011001010



The F function

To XOR with 48-bit 16th subkey, 32-bit RB needs to be 48-bit, by using permutation method (Expansion Permutation) to add extra bits.

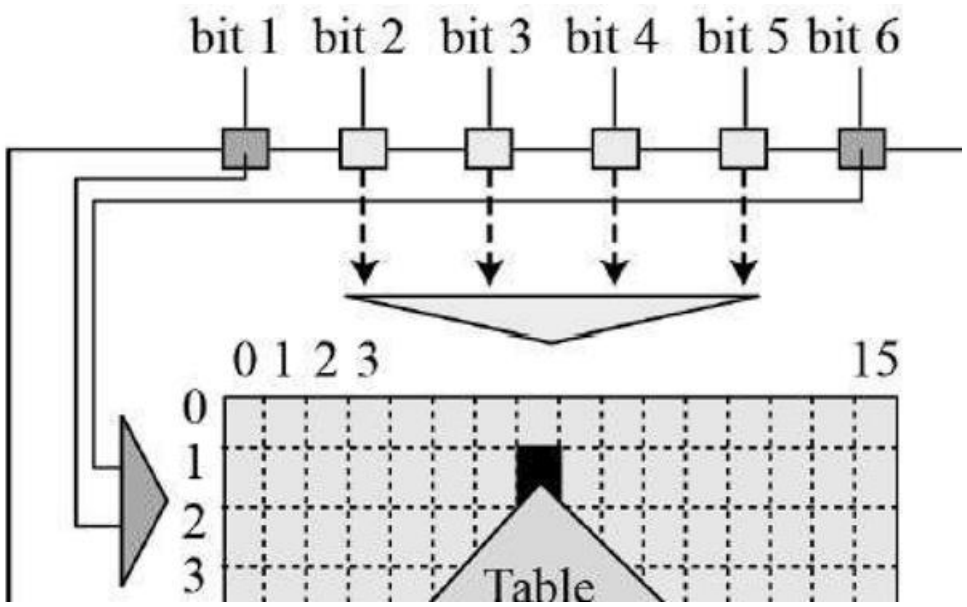


32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

Expansion Permutation

The value after XOR will be divided into 6-bit groups. Each group is 6 inputs of a S-box, which has 4 outputs.

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011



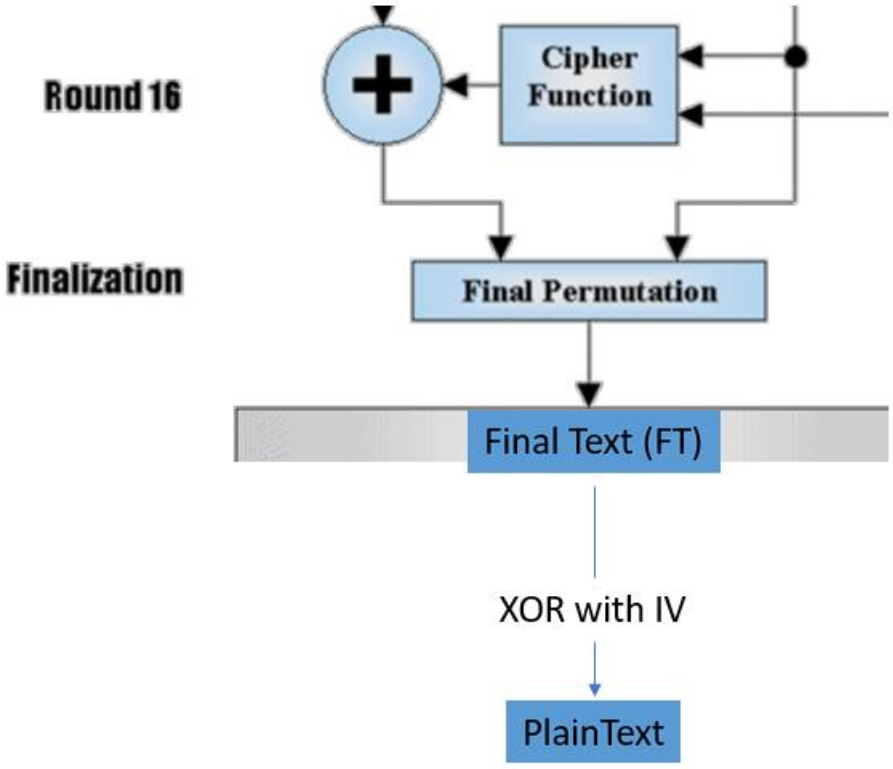
DES S-box logic table

For example a 6-bit input is 011011" has lsb 1, msb 0 and inner bits "1101" then output is "1001".

With 8 S-box then we have 8 . 4 = 32 bits, those will be RB of Round 2. Meanwhile, the RB of Round 1:

11000010111011101110010011001010

will be used as LB of Round 2. At the Round 16, LB and RB of that round will be merged to form the final text (FT). The first 64-bit plaintext block is the result of FT XOR IV.



First PlainText block is formed

The second 64-bit block of cipher is the ciphertext bits that are not used yet (11101110110011001100001), following by the padding bits (assume a padding bit is 0). The first 64-bit block will be XOR with the FT of the second decryption, instead of IV like the first decryption, to form the second plaintext block.

3. Exercise 3:

The <io.h> and <fcntl.h> library supports _setmode function, which sets the file translation mode. It can open the file in UTF-16 mode.

```
_setmode(_fileno(stdin), _O_U16TEXT);
string plain = "Hải Anh";
```

However, it is not working yet.

4. Exercise 4:

Change the default plaintext ("CBC Mode Test") into the code below:

```
string plain;
cout << "Nhập plaintext: ";
std::getline(std::cin, plain);
```

In my gcc (12.2), gdb (13.1), the last input (Enter, or '\n') has been automatically ignored, so it is not necessary to add cin.ignore function.

Other codes are the same.

5. **Exercise 5:**

Type key: The key is SecByteBlock type, a special Byte array of CryptoPP. Type a string named keystring and use reinterpret_cast to cast to a SecByteBlock object, named key.

```
string keystring;
cout << "Type 8-byte key: ";
std::getline(std::cin, keystring);
// Convert string  to SecByteBlock
// &keystring[0]: the beginning of the data
// keystring.size: the size of the array
SecByteBlock key(reinterpret_cast<const byte*>(&keystring[0]), keystring.size());
```

string to SecByteBlock

This code will not set the keystring length, so I must type 8 characters to fit the SecByteBlock length or an error is thrown out.

Type iv: It is possible to import a value to a byte element with cin. The input value is a char, and compiler will convert automatically to a byte.

```
byte iv[8];
for (size_t i = 0; i < 8; i++){
    cout << i << ": ";
    std::cin >> iv[i];
}
```

```
Type 8-byte key: abcdabcd
0: 1
1: 2
2: 3
3: 4
4: 5
5: 6
6: 7
7: 8
cipher text: ÷oÿ┐vVïe4iö¢¡S,
Recovered text: CBC Mode Test
```

example of import key and iv

The code does not encode cipher in Hex format, so it is not easy to look at.

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hành theo hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

- File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.
- Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).

  *Ví dụ: [*NT101.K11.ANTT*]-Session1_Group3.*
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

- Chuẩn bị tốt.
- Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT