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: 24/03/2023

1. THÔNG TIN CHUNG:

Lớp: NT219.N22.ATCL.1

STT	Họ và tên	MSSV	Email
1	Đoàn Hải Đăng	21520679	21520679@gm.uit.edu.vn
2	Lê Thanh Tuấn	21520518	21520518@gm.uit.edu.vn
3	Phan Thị Hồng Nhung	21521250	21521250@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Chậm lại và suy nghĩ	100%	Lê Thanh Tuấn
2	Bài tập 1	100%	Phan Thị Hồng Nhung
3	Bài tập 2	100%	Phan Thị Hồng Nhung
4	Bài tập 3	100%	Đoàn Hải Đăng
5	Bài tập 4	100%	Đoàn Hải Đăng
6	Bài tập 5	100%	Lê Thanh Tuấn
7	Differential cryptanalysis attack	60%	Đoàn Hải Đăng
8	Linear cryptanalysis attack	60%	Phan Thị Hồng Nhung
9	Bài luyện tập 1	80%	Đoàn Hải Đăng
10	Bài luyện tập 2	50%	Phan Thị Hồng Nhung
11	Bài luyện tập 3	70%	Lê Thanh Tuấn

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

BÁO CÁO CHI TIẾT

1. Chậm lại và suy nghĩ

Permuted choice 1 đã làm gì để từ 64-bit key thành 56-bit key?

Answer: The initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded for parity checking. Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key.

DES::DEFAULT_KEYLENGTH và DES::BLOCKSIZE bằng bao nhiêu?

Answer: Both are 8 bytes.

```
cout << "key length: " << DES::DEFAULT_KEYLENGTH << endl;
cout << "block size: " << DES::BLOCKSIZE << endl;
```

```
key length: 8
block size: 8
key: DBCAEEA01F5E0FF8
iv: F44FBC37EB6AB552
plain text: CBC Mode Test
cipher text: BBF6B653A2C575FC6B81C84861D4BDD3
recovered text: CBC Mode Test
```

2. Bài tập 1 :

☞ *Debug the program and describe the encryption progress of DES using CBC mode.*

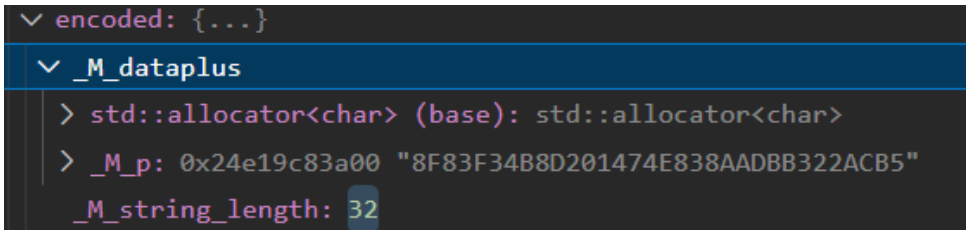
```
key: {...}
  m_alloc
  m_mark: 18446744073709551615
  m_size: 8
  m_ptr: 0x24e19e31600 "\002\203
iv: [8]
  [0]: 225 '\341'
  [1]: 29 '\035'
  [2]: 28 '\034'
  [3]: 138 '\212'
  [4]: 108 'l'
  [5]: 181 '\265'
  [6]: 149 '\225'
  [7]: 192 '\300'
```

- IV and key are initialized randomly by the program.

```
SecByteBlock key(DES::DEFAULT_KEYLENGTH);
prng.GenerateBlock(key, key.size());

CryptoPP::byte iv[DES::BLOCKSIZE];
prng.GenerateBlock(iv, sizeof(iv));
```

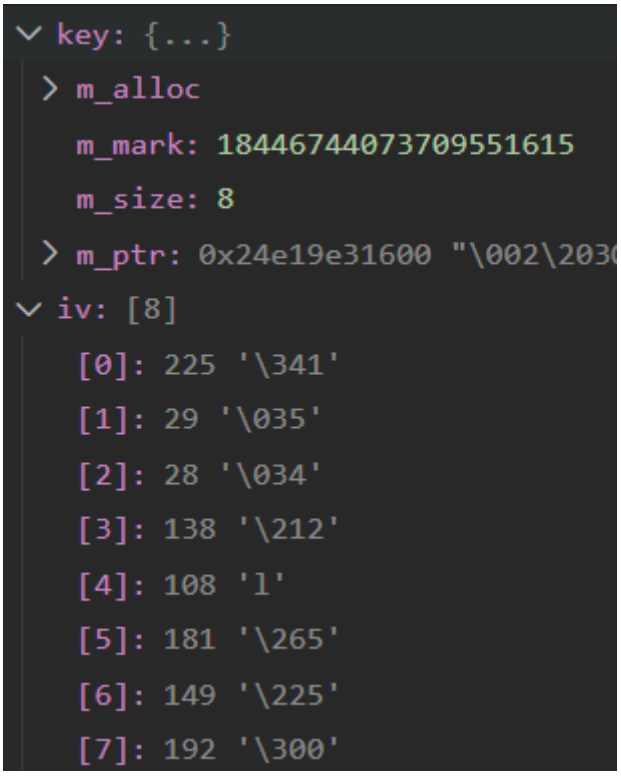
- Ciphertext is generated after the encryption progress.

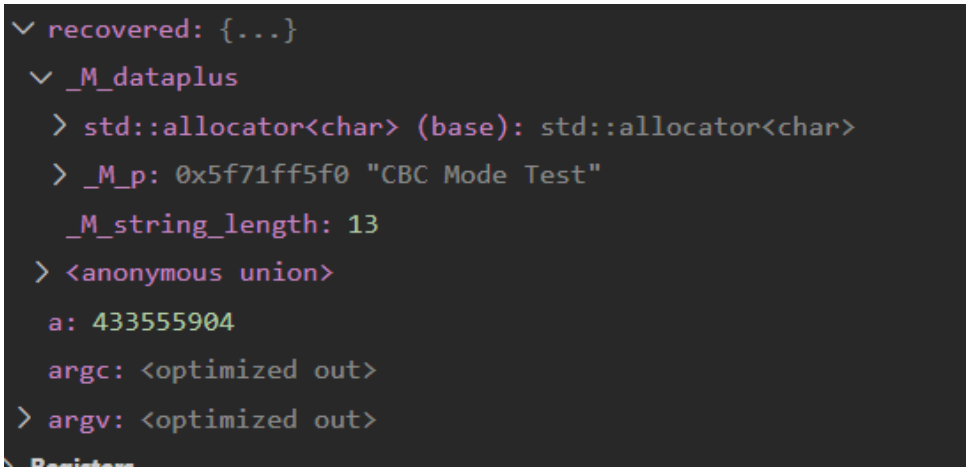
```
    CBC_Mode< DES >::Encryption e;
    e.SetKeyWithIV(key, key.size(), iv);

    // The StreamTransformationFilter adds padding
    //  as required. ECB and CBC Mode must be padded
    //  to the block size of the cipher.
    StringSource(plain, true,
        new StreamTransformationFilter(e,
            new StringSink(cipher)
        ) // StreamTransformationFilter
    ); // StringSource
```



3. Bài tập 2

☞ *Debug the program and describe the decryption progress of DES using CBC mode.*





- Decryption reuses the key and IV in the Encryption process to decrypt.
- Then we have the input recovered.

4. Bài tập 3

☞ *Plaintext supports input including UTF-16 characters.*

- Key and iv are randomly generated.
- Set the plain text = "*thử nghiệm tiếng Việt*", the cipher text after encryption="33C7307D395769F9AB715A987CC6F082EAB1B929A22A8E4D0D21EF0648AAF094".
- After decryption, the program gives recovered text = plain text.

```
PS D:\Selfwork\Crypto> .\lab1_DES_CBC_ex3.exe
key: 1726D04371740108
iv: 52227E22CC3117C7
plain text: thử nghiệm tiếng Việt
cipher text: 33C7307D395769F9AB715A987CC6F082EAB1B929A22A8E4D0D21EF0648AAF094
recovered text: thử nghiệm tiếng Việt
```

5. Bài tập 4

☞ *Plaintext is manually entered into the program by user.*

- In this case, we enter input = "*thử bài này nè thầy*", key and iv are randomly generated by the program.
- After decryption, the program gives recovered text = input.

```
PS D:\Selfwork\Crypto> .\lab1_DES_CBC_ex4.exe
Enter input: thử bài này nè thầy
key: A43C404F0B2349A6
iv: CAA545C78A872A6B
plain text: thử bài này nè thầy
cipher text: 0A52C90D6523673B3D4FAB73FBCD050936DB8413FA474E05493682D5F4969314
recovered text: thử bài này nè thầy
```

6. Bài tập 5

☞ *Secret key and IV are manually entered into the program by user.*

- Enter key and iv (8 bits).
- The plain text is set = "*Hồng Nhung 1234567890*".
- The program prints out the cipher text and recovered text correctly.

```
PS D:\Selfwork\Crypto> .\lab1_DES_CBC_ex5.exe
Enter key (8 bits): 12341234
Enter iv (8 bits): 32323232
key: 3132333431323334
iv: 3332333233323332
plain text: Hồng Nhung 1234567890
cipher text: 7A569B194A289C785D42F3B8C5EDAA951026CD1C7DBE6354
recovered text: Hồng Nhung 1234567890
```

7. Differential cryptanalysis attack

- One of the most significat recent (public) advances in cryptanalysis.
- Powerful method to analyse block ciphers.
- Differential Cryptanalysis compares two related pairs of encryptions which can leak information about the key, given a sufficiently large number of suitable pairs.

- With a known difference in the input searching for a known difference in output. So it is called Differential Cryptanalysis.

8. Linear cryptanalysis attack

- This attack is based on finding linear approximations to describe the transformations performed in DES.
- This method can find a DES key given 2^43 known plaintexts, as compared to 2^47 chosen plaintexts for differential cryptanalysis.
- The objective of linear cryptanalysis is to find an effective linear equation relating some plaintext, ciphertext and key bits that holds with probability p != 0.5 . P[i_1, i_2 , ...,i_a] © C[j_1, j_2, ..., j_b] = K[k_1,k_2,...,k] where ijk are bit locations in P,C,K.
- Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the equation for a large number of plaintext-ciphertext pairs.

9. Bài luyện tập 1:

☞ *Evaluate DES algorithm with CBC mode.*

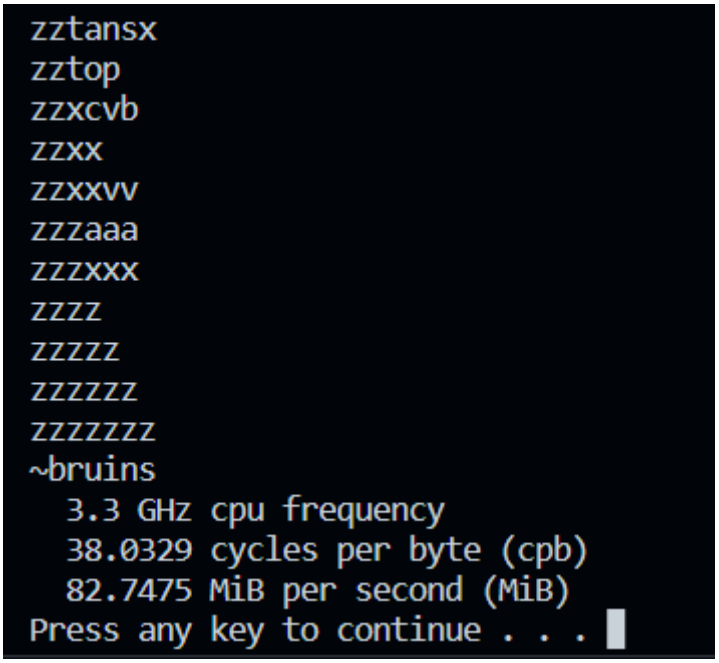
- Case 1: Data smaller than 64 bits.

```
key: 92576A305AC195D8
iv: B6E70693AF8715EB
cipher text: 7E5F0BF07687A74B
recovered text: abf451s
  3.3 GHz cpu frequency
  37.2645 cycles per byte (cpb)
  84.4536 MiB per second (MiB)
Press any key to continue . . .
```

- Case 2: Data in UTF-16 format.

```
key: 0DE4C1B7FC292A73
iv: B2C58A10959E12AB
cipher text: AB6071CA90051649915576AF584511CA12E4538B1DDAF949
  3.3 GHz cpu frequency
  36.7843 cycles per byte (cpb)
  85.5561 MiB per second (MiB)
```

- Case 3: Data bigger than 1MB.



10. Bài luyện tập 2:

☞ *The disadvantages of DES algorithms:*

- Key length: The key length of DES is only 56 bits, which is considered insufficient for modern cryptographic standards. With modern computing power, it is possible to brute-force attack a DES key and recover the plaintext in a reasonable amount of time.
- Vulnerability to attacks: DES is vulnerable to several attacks, including brute-force attacks, differential cryptanalysis, and linear cryptanalysis. These attacks can be used to recover the key and plaintext.
- Limited block size: The block size of DES is only 64 bits, which can lead to security vulnerabilities. In particular, DES is vulnerable to attacks that exploit the limited block size, such as the birthday attack.
- Limited security: DES provides only moderate security, and it is not considered sufficient for use in high-security applications. It has been recommended that DES be used only for legacy systems and not for new systems.
- Restricted usage: The use of DES is restricted by export controls, making it difficult to use in international applications. Additionally, the use of DES is not compliant with some security regulations, such as PCI DSS.

☞ *The attack logic*

- Initialize a counter variable and an empty result variable. Divide the plaintext into blocks of the same length as the key (for example, if the key is 128 bits long, the plaintext is also divided into blocks of 128 bits).

- For each plaintext block, use the encryption algorithm to encrypt the block with the given IV and key.
- Compare the encrypted block with known values of plaintext. If the encrypted block matches any of the known plaintext values, save the block to the result variable. Increase the counter variable by 1 and go back to step 3 until all plaintext blocks have been encrypted.
- After the program ends, the result variable will contain the discovered plaintext blocks. However, the ability to find the correct plaintext depends on the number of known plaintext values and the runtime of the program. If the number of known plaintext values is high and the runtime of the program is not too long, the likelihood of finding the correct plaintext is high.

11. Bài tập luyện tập 3:

☞ *Weaknesses of CBC.*

- Can not process data blocks concurrently: CBC cannot process data blocks concurrently, so the encryption and decryption speed of CBC may be slower than other modes.
- Possibility of being attacked by pseudo-encrypted coils.
- Initialization vector dependency

```cpp
#include <iostream>
#include <string>
#include <cstring>
#include <openssl/aes.h>

using namespace std;

string break_cbc(string iv, string ciphertext, string key) {
    int block_size = AES_BLOCK_SIZE;
    AES_KEY aes_key;
    AES_set_decrypt_key((const unsigned char*)key.c_str(), block_size*8, &aes_key);
    string plaintext = "";
    for (int i = 0; i < ciphertext.length(); i += block_size) {
        string block = ciphertext.substr(i, block_size);
        string previous_block = (i == 0) ? iv : ciphertext.substr(i-block_size, block_size);
        string decrypted_block = "";
        AES_decrypt((const unsigned char*)block.c_str(), (unsigned char*)decrypted_block.c_str(), &aes_key);
        for (int j = 0; j < block_size; j++) {
            plaintext += (char)(decrypted_block[j] ^ previous_block[j]);
        }
    }
    return plaintext;
}
```

```
int main() {
    string iv = "0123456789abcdef";
    string key = "abcdefghijklmnop";
    string plaintext = "This is a test message.";
    int block_size = AES_BLOCK_SIZE;

    // Encrypt the plaintext using CBC mode
    AES_KEY aes_key;
    AES_set_encrypt_key((const unsigned char*)key.c_str(), block_size*8, &aes_key);
    int padded_len = ((plaintext.length() + block_size - 1) / block_size) * block_size;
    string padded_plaintext = plaintext;
    if (padded_len > plaintext.length()) {
        padded_plaintext.resize(padded_len, '\0');
    }
    string ciphertext = iv;
    for (int i = 0; i < padded_len; i += block_size) {
        string block = padded_plaintext.substr(i, block_size);
        string previous_block = ciphertext.substr(i, block_size);
        for (int j = 0; j < block_size; j++) {
            block[j] ^= previous_block[j];
        }
        string encrypted_block = "";
        AES_encrypt((const unsigned char*)block.c_str(), (unsigned char*)encrypted_block.c_str(), &aes_key);
        ciphertext += encrypted_block;
    }
```

```
51      // Break the CBC encryption using the attack
52      string broken_plaintext = break_cbc(iv, ciphertext, key);
53      cout << "Broken plaintext: " << broken_plaintext << endl;
54
55      return 0;
56  }
```

The attack logic

- The program shown above has two main parts: the CBC encryption part and the CBC attack using pseudo-encrypted rollover.
- The CBC encryption part begins with generating an AES key from the provided key using the AES_set_encrypt_key function. It then adds an initialization vector (IV) to the top of the ciphertext and encrypts each block of plaintext using CBC mode. To do this, it must take a plaintext block, XOR it with the previous ciphertext block or initialization vector (if it's the first), and then encrypt the XOR block using the AES_encrypt function. Finally, it appends the cipher block to the ciphertext.
- The CBC attack begins with three input parameters: the initialization vector, the ciphertext, and the key. First, it generates an AES key from the provided key using the AES_set_decrypt_key function. Then it starts processing the ciphertext block by block. For each block, it decrypts the block using the AES_decrypt function to find the original block. It then XORs the original block with the previous ciphertext block or initialization vector (if it's the first one) to find the plaintext block. Finally, it appends the plaintext block to the plaintext under construction.

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