ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP NHẬP MÔN AN TOÀN THÔNG TIN

Programming Assigment 2 : Lập trình mô phỏng
TRIVIUM STREAM CIPHER

NGUYỄN HỮU DŨNG
Mã số sinh viên: 20215545

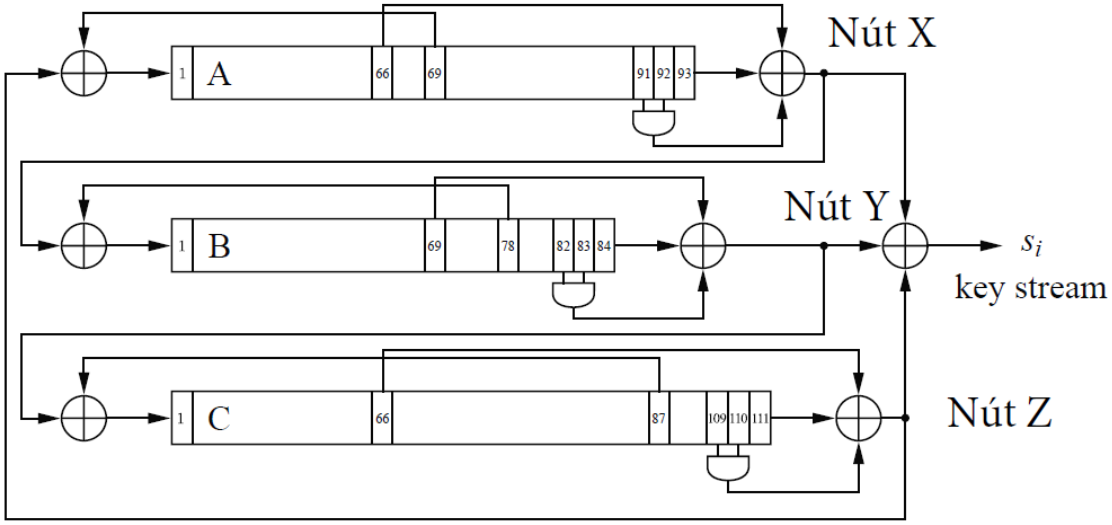
Mã lớp: 149480

Giảng viên : TS. Trần Vĩnh Đức

HÀ NỘI, 3/2024

I. Tóm tắt đề bài:

1. Cấu tạo:



Trivium là hệ mã dòng với kích thước khoá 80 bit. Dựa trên việc kết hợp ba thanh ghi dịch có phản hồi, và kết hợp với thành phầnn phi tuyến. Ba thanh ghi A, B, C với kích thước mỗi thanh ghi chứa được lần lượt 93 bits, 84 bits, 111 bits.

Ban đầu, người dùng đưa vào 2 input là IV (80 bit) và key (80 bit). Trạng thái của từng thanh ghi như sau:

- 80 bit IV được đưa vào 80 bit trái nhất của thanh ghi A.
- 80 bit khoá được đưa vào 80 bit trái nhất của thanh ghi B.
- Mọi bit thanh ghi C có giá trị 0 ngoại trừ ba bit phải nhất :

$$c_{109} = c_{110} = c_{111} = 1$$

2. Hoạt động:

- Pha khởi động:

Trong pha đầu tiên này, hệ mã được chạy 4 x (93 + 84 + 111) = 1152 lần, nhưng không tạo ra bit đầu ra nào. Pha này cần để tạo cho hệ mã đủ ngẫu nhiên, đảm bảo key stream phụ thuộc vào cả key và IV.

- Pha sinh khóa:

Dãy bit sau đó bắt đầu từ chu kỳ 1153, được sử dụng như dòng khoá s_i của hệ mã dòng.

II. Mô phỏng chương trình:

Ngôn ngữ lập trình: C++

Tên file chương trình: Trivium.cpp

1. Khởi tạo:

```
//a, b, c đại diện cho 3 thanh ghi
deque<char> a;
deque<char> b;
deque<char> c;

string IV, key;
```

Ba thanh ghi được mô phỏng bằng 3 deque, thuận tiện mô phỏng việc thêm bit mới (khi thêm 1 bit mới vào đầu là push_front(), và pop_back() nếu thừa bit).

Mỗi phần tử trong thanh ghi có giá trị 0 hoặc 1 (để kiểu char cho tiết kiệm bộ nhớ).

Input là IV và key là 2 đầu vào có kích thước 80 bit.

2. Pha khởi động:

```
void warm_up() {
    //Thiết lập các bit trên 3 thanh ghi như mô tả đề bài:
    for(int i = 0; i < 80; i++) {
            a.push_back(IV[i] - '0');
            b.push_back(key[i]- '0');
    }
    c.assign(111, 0);
    c[108] = 1;
    c[109] = 1;
    c[110] = 1;

    /*1152 chu kì đầu đưa thêm bit vào các thanh ghi như mô tả nhưng chưa sinh ra bit nào của stream key
    Những phép Xor nào mà chưa đủ thành phần thì không làm, chuyển qua chu kì tiếp theo
    Khi nào thanh ghi vượt quá số bit cho phép thì mới pop_back() ra
    */
    for (int i = 0; i < 1152; i++){
        a.push_front(c[65] ^ c[110] ^ (c[108]&&c[109]) ^ a[68]);
        if (a.size() > 93) a.pop_back();
        if (a.size() == 93) {
            b.push_front(a[65] ^ a[92] ^ (a[90] && a[91]) ^ b[77]);
            if (b.size() > 84) b.pop_back();
        }
        if (b.size() == 84) {
            c.push_front(b[68] ^ b[83] ^ (b[81] && b[82]) ^ c[86]);
            c.pop_back();
        }
    }
}
```

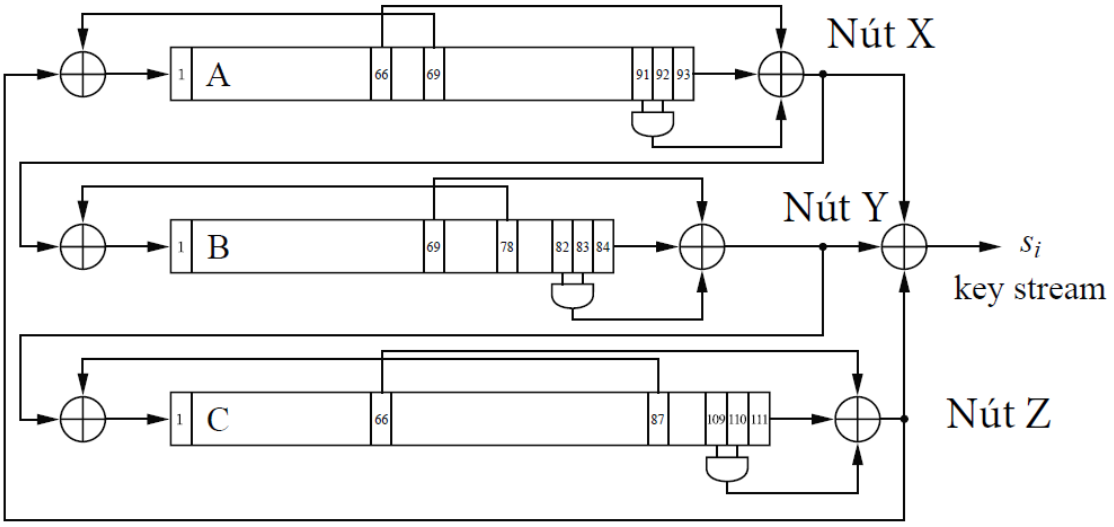
3. Sinh Stream Key:

```
vector<char> encryption(int size) {
    vector<char> stream_key; //Chứa dãy bit stream key được sinh ra
    for (int i = 0; i < size; i++){
        //x, y, z lần lượt là giá trị tại 3 nút X, Y, Z trên hình
        char x = a[65] ^ a[92] ^ (a[90] && a[91]);
        char y = b[68] ^ b[83] ^ (b[81] && b[82]);
        char z = c[65] ^ c[110] ^ (c[108]&&c[109]);
        //Các bit được sinh ra để output và đưa vào thanh ghi theo mô tả
        stream_key.push_back(x ^ y ^ z);

        a.push_front(z ^ a[68]);
        a.pop_back();

        b.push_front(x ^ b[77]);
        b.pop_back();

        c.push_front(y ^ c[86]);
        c.pop_back();
    }
    return stream_key;
}
```



4. Mã hóa file:

```
void cipher(string read, string write) {
    //Mở file bằng ifstream, ofstream
    ifstream input_file(read);
    ofstream output_file(write);
    string line;
    //Đọc từng dòng của file input
    while (getline(input_file, line)){
        vector<char> stream_key = encryption(line.size()*8); //Sinh key với chiều dài tương ứng
        vector<char> bin_content = string_to_binary(line); //Chuyển text sang chuỗi bit
        vector<char> encrypted = xor_two_binary(bin_content, stream_key); //Xor chúng với nhau
        string result = bin_to_text(encrypted); //Chuyển chuỗi bit sang text
        output_file << result << "\n"; //Ghi ra file output (kèm ký tự xuống dòng)
    }
    //Đóng file
    input_file.close();
    output_file.close();
}
```

Trong đó:

- Hàm chuyển text sang chuỗi bit theo bảng mã ASCII:

```
vector<char> string_to_binary(const string &content) {
    vector<char> bits;
    for (char c : content) {
        // Chuyển ký tự thành mã ASCII
        int asciiValue = static_cast<int>(c);

        // Chuyển mã ASCII thành dãy bit và đưa vào vector
        bitset<8> bitset(asciiValue);
        for (int i = 7; i >= 0; --i) {
            bits.push_back(bitset[i]);
        }
    }
    return bits;
}
```

- Hàm XOR 2 dãy bit:

```
vector<char> xor_two_binary(const vector<char>& bits1,
                            const vector<char>& bits2) {
    vector<char> result;
    size_t size = bits1.size();

    for (size_t i = 0; i < size; ++i) {
        result.push_back(bits1[i] ^ bits2[i]);
    }
    return result;
}
```

- Hàm chuyển dãy bit sang text:

```
string bin_to_text(const vector<char> &bits) {
    string ascii;
    //Đọc từng 8 bit một, chuyển tương đương sang 1 ký tự theo bảng mã ASCII
    for (int i = 0; i < bits.size(); i += 8) {
        char byte = 0;
        for (int j = 0; j < 8; j++) {
            byte |= (bits[i + j] << (7 - j));
        }
        ascii += byte;
    }
    return ascii;
}
```

III. Kết quả:

Khai báo mảng động chứa tên các file đọc và ghi tương ứng:

```
vector<char*>inputs = {"alice29.txt",
                       "asyoulik.txt",
                       "bible.txt",
                       "cp.html",
                       "E.coli",
                       "fields.c",
                       "grammar.lsp",
                       "kennedy.xls",
                       "lcet10.txt",
                       "plrabn12.txt",
                       "ptt5",
                       "sum",
                       "world192.txt",
                       "xargs.1"};
vector<char*>outputs = {"out_alice29.txt",
                       "out_asyoulik.txt",
                       "out_bible.txt",
                       "out_cp.html",
                       "out_E.coli",
                       "out_fields.c",
                       "out_grammar.lsp",
                       "out_kennedy.xls",
                       "out_lcet10.txt",
                       "out_plrabn12.txt",
                       "out_ptt5",
                       "out_sum",
                       "out_world192.txt",
                       "out_xargs.1"};
```

Hàm main() sẽ tiến hành mã hóa từng file:

```
int main() {
    /*Lặp qua từng cặp file input và output
    Với từng file, người dùng nhập vào chuỗi bit IV và key tương ứng*/
    for (int i = 0; i < inputs.size(); i++) {
        cout << "Enter IV for file \"" << inputs[i] << "\"\n";
        cin >> IV;
        cout << "Enter Key for file \"" << inputs[i] << "\"\n";
        cin >> key;
        //Khởi động sau đó tiến hành mã hóa
        warm_up();
        cipher(inputs[i], outputs[i]);
        cout << "File encrypted : \"" << outputs[i] << "\"\n";
    }

    return 0;
}
```

Giả sử tất cả các IV và key để mã hóa đều như nhau:

IV="10101010110011001111000011110000111100001111000011110000111100001111000011
11000011110000";

key="11001100011110000011110000111100001111000011110000111100001111000011110000111100001111000011110000";

```
int main() {
    IV = "10101010110011001111000011110000111100001111000011110000111100001111000011110000111100001111000011110000";
    key = "11001100011110000011110000111100001111000011110000111100001111000011110000111100001111000011110000111100";

    /*Lặp qua từng cặp file input và output
    Với từng file, người dùng nhập vào chuỗi bit IV và key tương ứng*/
    for (int i = 0; i < inputs.size(); i++) {
//          cout << "Enter IV for file \"" << inputs[i] << "\"\n";
//          cin >> IV;
//          cout << "Enter Key for file \"" << inputs[i] << "\"\n";
//          cin >> key;
        //Khởi động sau đó tiến hành mã hóa
        warm_up();
        cipher(inputs[i], outputs[i]);
        cout << "File encrypted : \"" << outputs[i] << "\"\n";
    }

    return 0;
}
```

Kết quả:

```
File encrypted : "out_alice29.txt"
File encrypted : "out_asyoulik.txt"
File encrypted : "out_bible.txt"
File encrypted : "out_cp.html"
File encrypted : "out_E.coli"
File encrypted : "out_fields.c"
File encrypted : "out_grammar.lsp"
File encrypted : "out_kennedy.xls"
File encrypted : "out_lcet10.txt"
File encrypted : "out_plrabn12.txt"
File encrypted : "out_ptt5"
File encrypted : "out_sum"
File encrypted : "out_world192.txt"
File encrypted : "out_xargs.1"

Process returned 0 (0x0)   execution time : 22.254 s
```