Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине «Криптографические методы защиты информации»

Тема: «Симметричные криптоалгоритмы»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Литвинюк Т. В.

**Проверил:**

Хацкевич М. В.

Брест 2023

**Цель:** научиться создавать алгоритмы симметричного шифрования.

**Ход работы:**

**Вариант 6**

6. Перестановка OFB 11 байт Число подблоков 4

**Задание**

Разработать собственный алгоритм который реализует указанный в варианте:

режим использования блочного шифра;

работает с указанной длиной блока;

позволяет оценивать скорость шифрования/дешифрования.

#include "../Libs/bitset.h"

#include "../Libs/vector.h"

#include <iostream>

using namespace std;

vector<unsigned> get\_order(vector<bitset<16>> &key){

// Получить порядок {4, 5, 2, 7}

vector<unsigned> shadow, order(key.size());

for(bitset<16> &el: key){

shadow.push\_back(el.to\_int());

}

unsigned index, cur\_num;

for(int t = 0; t < key.size(); t++){

cur\_num = -1;

for(int i = 0; i < key.size(); i++){

if (cur\_num >= shadow[i]){

cur\_num = shadow[i];

index = i;

}

}

order[index] = t;

shadow[index] = -1;

}

return order;

}

vector<bitset<16>> get\_bitset(string &str){

vector<bitset<16>> set;

for (char symb: str)

set.push\_back(bitset<16>(symb));

return set;

}

string get\_string(vector<bitset<16>> &set){

string str;

for(bitset<16> &el: set)

str += el.to\_int();

return str;

}

vector<bitset<16>> permute\_encrypt(vector<bitset<16>> &plaintext, vector<bitset<16>> &key, bool showInfo=true){

// Шифрование перестановками

int rows = plaintext.size() / key.size(), columns = key.size();

vector<vector<bitset<16>>> tabl(rows, vector<bitset<16>>(columns));

vector<unsigned> order = get\_order(key);

// заполнение tabl

for (int x = 0; x < plaintext.size(); x++){

tabl[x % rows][x / rows] = plaintext[x];

}

if (showInfo)

cout << "33%";

// перестановка столбцов согласно order

vector<vector<bitset<16>>> tabl2(rows, vector<bitset<16>>(columns));

for(int j = 0; j < order.size(); j++){

for(int i = 0; i < rows; i++){

tabl2[i][order[j]] = tabl[i][j];

}

}

if (showInfo)

cout << "\b\b\b66%";

vector<bitset<16>> ciphertext;

for (int i = 0; i < rows; i++){

for (int j = 0; j < columns; j++){

ciphertext.push\_back(tabl2[i][j]);

}

}

if (showInfo)

cout << "\b\b\b100%" << endl;

return ciphertext;

}

vector<bitset<16>> permute\_decrypt(vector<bitset<16>> &ciphertext, vector<bitset<16>> &key, bool showInfo=true){

// Расшифрование перестановками

int rows = ciphertext.size() / key.size(), columns = key.size();

vector<vector<bitset<16>>> tabl(rows, vector<bitset<16>>(columns));

vector<unsigned> order = get\_order(key);

// заполнение tabl

for (int x = 0; x < ciphertext.size(); x++){

tabl[x / columns][x % columns] = ciphertext[x];

}

if (showInfo)

cout << "33%" << endl;

// перестановка столбцов обратно согласно order

vector<vector<bitset<16>>> tabl2(rows, vector<bitset<16>>(columns));

for(int j = 0; j < order.size(); j++){

for(int i = 0; i < rows; i++){

tabl2[i][j] = tabl[i][order[j]];

}

}

if (showInfo)

cout << "\b\b\b66%" << endl;

vector<bitset<16>> plaintext;

for (int i = 0; i < columns; i++){

for (int j = 0; j < rows; j++){

plaintext.push\_back(tabl2[j][i]);

}

}

if (showInfo)

cout << "\b\b\b100%" << endl;

return plaintext;

}

vector<bitset<16>> xor\_(vector<bitset<16>> &vec1, vector<bitset<16>> &vec2){

vector<bitset<16>> result(vec1.size());

for (int i = 0; i < vec1.size(); i++){

result[i] = vec1[i] ^ vec2[i];

}

return result;

}

template <int block\_size=11, int block\_count=4>

vector<bitset<16>> ofb(vector<bitset<16>> &plaintext, vector<bitset<16>> &key, vector<bitset<16>> &iv, bool showInfo=true){

vector<bitset<16>> ciphertext, prev\_block = iv, prev\_block2, block\_xor; int i;

if (showInfo)

cout << "00%";

for(i = 0; i < plaintext.size() - block\_size + 1; i += block\_size){

prev\_block2 = permute\_encrypt(prev\_block, key, false);

vector<bitset<16>> block(plaintext, i, block\_size);

block\_xor = xor\_(block, prev\_block2);

ciphertext.insert(block\_xor);

prev\_block = prev\_block2;

if (showInfo)

cout << "\b\b\b" << (int)(i / plaintext.size() \* 100) << "%";

}

if (plaintext.size() % block\_size != 0){

// int last\_block\_size = plaintext.size() % block\_size;

// vector<bitset<16>> last\_block(block\_size);

// copy(plaintext.end()-last\_block\_size, plaintext.end(), last\_block.begin());

// prev\_block2 = permute\_encrypt(prev\_block, key, false);

// block\_xor = xor\_(last\_block, prev\_block2);

// ciphertext.insert(block\_xor);

ciphertext.insert(plaintext.subvector(i, plaintext.size() % block\_size));

}

if (showInfo)

cout << "\b\b\b100%" << endl;

return ciphertext;

}

**Вывод:** в ходе лабораторной работы я научился шифровать и сжимать информацию.