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-23

Романюк А.П.

**Проверил:**

Хацкевич А. С.

Брест 2024

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

**Ход работы:**

**Вариант 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Матричный метод | CBC | 9 байт | - |

**lib.h**

#pragma once

#include <iostream>

#include <fstream>

#include <cmath>

#include <vector>

#include <string>

#include <sstream>

#include "matrix.h"

using namespace std;

string padding(string word, int n\_bits) {

while (word.size() % n\_bits != 0) {

word += '.';

}

return word;

}

vector<int> XOR(vector<int> a, vector<int> b) {

vector<int> result(a.size());

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

result[i] = (a[i] ^ b[i])%256;

}

return result;

}

vector<int> matrixshifr(vector<int> block, vector<int> key,int n\_bits) {

int pos = 0;

vector<vector<int>> m\_block(sqrt(n\_bits),vector<int>(sqrt(n\_bits)));

vector<vector<int>> m\_key(sqrt(n\_bits), vector<int>(sqrt(n\_bits)));

for (int i = 0; i < sqrt(n\_bits); i++) {

for (int j = 0; j < sqrt(n\_bits); j++) {

m\_block[i][j] = block[pos];

m\_key[i][j] = key[pos];

pos++;

}

}

vector<vector<int>> m\_result(sqrt(n\_bits), vector<int>(sqrt(n\_bits)));

for (int i = 0; i < sqrt(n\_bits); ++i) {

for (int j = 0; j < sqrt(n\_bits); ++j) {

for (int k = 0; k < sqrt(n\_bits); ++k) {

m\_result[i][j] += m\_key[i][k] \* m\_block[k][j];

}

}

}

vector<int> result;

for (const auto& row : m\_result) {

for (int val : row) {

result.push\_back(val);

}

}

/\*cout << "cipher block\n";

for (const auto& c : result) {

cout << c << " ";

}

cout << '\n';\*/

return result;

}

vector<int> unmatrixshifr(vector<int> cipherblock, vector<int> key, int n\_bits) {

int pos = 0;

vector<vector<int>> m\_cipherblock(sqrt(n\_bits), vector<int>(sqrt(n\_bits)));

vector<vector<double>> m\_key(sqrt(n\_bits), vector<double>(sqrt(n\_bits)));

for (int i = 0; i < sqrt(n\_bits); i++) {

for (int j = 0; j < sqrt(n\_bits); j++) {

m\_cipherblock[i][j] = cipherblock[pos];

m\_key[i][j] = key[pos];

pos++;

}

}

m\_key = inverseMatrix(m\_key);

vector<vector<int>> m\_result(sqrt(n\_bits), vector<int>(sqrt(n\_bits)));

for (int i = 0; i < sqrt(n\_bits); ++i) {

for (int j = 0; j < sqrt(n\_bits); ++j) {

for (int k = 0; k < sqrt(n\_bits); ++k) {

m\_result[i][j] += m\_key[i][k] \* m\_cipherblock[k][j];

}

}

}

vector<int> result;

for (const auto& row : m\_result) {

for (int val : row) {

result.push\_back(val);

}

}

/\*cout << "uncipher block\n";

for (const auto& c : result) {

cout << c << " ";

}

cout << '\n';\*/

return result;

}

vector<int> CBC(string text, vector<int> key, int n\_bits) {

text = padding(text, n\_bits);

vector<int> iv = key;

vector<int> ciphertext;

for (int iter = 0; iter < text.size(); iter += n\_bits) {

vector<int> block;

for (int i = iter; i < iter+n\_bits; ++i) {

block.push\_back(text[i]);

}

block = XOR(block, iv);

//тут будет функция шифрования

vector<int> cipherblock = matrixshifr(block,key,n\_bits);

ciphertext.insert(ciphertext.end(), cipherblock.begin(), cipherblock.end());

iv = cipherblock;

}

/\*for (auto c : ciphertext) {

cout << c << " ";

}

cout << endl;\*/

return ciphertext;

}

string unCBC(vector<int> ciphertext, vector<int> key, int n\_bits) {

vector<int> iv = key;

vector<int> text;

for (int iter = ciphertext.size()-1; iter > 0; iter-=n\_bits) {

vector<int> block;

for (int i = iter - n\_bits+1; i < iter+1; ++i) {

block.push\_back(ciphertext[i]);

}

vector<int> prevblock;

if (iter > n\_bits) {

for (int i = iter - 2 \* n\_bits+1; i < iter - n\_bits+1; ++i) {

prevblock.push\_back(ciphertext[i]);

}

}

else {

prevblock = iv;

}

//тут должна быть функция расшифра

block = unmatrixshifr(block,key,n\_bits);

vector<int> decryptedtext = XOR(block,prevblock);

text.insert(text.begin(), decryptedtext.begin(), decryptedtext.end());

}

string result = "";

for (auto c : text) {

result += c;

}

return result;

}

**encrypt.cpp**

#include "lib.h"

using namespace std;

int main() {

while (1) {

cout << "\_-\_main menu\_-\_\n";

cout << "1.Зашифровать текст\n";

cout << "2.Расшифровать текст\n";

cout << "3.Тестовый вариант\n";

cout << "4.Выход\n";

int choose;

cin >> choose;

switch (choose) {

case 1: {

system("cls");

string filename;

cout << "Введите название файла для шифрования\n";

cin >> filename;

ifstream inputFile(filename);

string text, key\_str;

getline(inputFile, text);

getline(inputFile, key\_str);

vector<int> key;

istringstream iss(key\_str);

int num;

while (iss >> num) {

key.push\_back(num);

}

ofstream outputFile("encrypted\_"+filename);

vector<int> encrypted\_text = CBC(text, key, 9);

for (auto c : CBC(text, key, 9)) {

outputFile << c << " ";

}

outputFile << endl;

for (auto c : key) {

outputFile << c << " ";

}

cout << "file " << "encrypted\_" + filename << " created";

outputFile.close();

break;

}

case 2: {

system("cls");

string filename;

cout << "Введите название файла для расшифрования\n";

cin >> filename;

ifstream inputFile(filename);

string text, key\_str;

getline(inputFile, text);

getline(inputFile, key\_str);

vector<int> ciphertext;

vector<int> key;

istringstream iss(key\_str);

istringstream txt(text);

int num;

while (iss >> num) {

key.push\_back(num);

}

while (txt >> num) {

ciphertext.push\_back(num);

}

ofstream outputFile("decrypted\_"+filename);

cout << "file "<<"decrypted\_"+filename <<" created\n";

outputFile << unCBC(ciphertext, key, 9);

outputFile << endl;

for (auto c : key) {

outputFile << c << " ";

}

break;

}

case 3: {

system("cls");

ifstream inputFile("test\_text.txt");

string text,key\_str;

getline(inputFile, text);

getline(inputFile, key\_str);

vector<int> key;

istringstream iss(key\_str);

int num;

while (iss >> num) {

key.push\_back(num);

}

ofstream outputFile("encrypted\_test\_text.txt\n");

vector<int> encrypted\_text = CBC(text, key, 9);

for (auto c : CBC(text, key, 9)) {

outputFile << c << " ";

}

outputFile << endl;

for (auto c : key) {

outputFile << c << " ";

}

cout << "file encrypted\_test\_text.txt created\n";

outputFile.close();

outputFile.open("decrypted\_test\_text.txt");

outputFile << unCBC(encrypted\_text, key, 9);

outputFile << endl;

for (auto c : key) {

outputFile << c << " ";

}

cout << "file decrypted\_test\_text.txt created\n";

break;

}

case 4:

system("cls");

return 0;

break;

default:

system("cls");

cout << "Такого варианта нет\n";

break;

}

}

}

Исходный текст: Дешифрованный текст

** **

Зашифрованный текст: :

****

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