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

Макаревич Н.Р.

Проверил:

Хацкевич А. С.

Брест 2024

Задание:

1. Изучить блочные алгоритмы шифрования: алгоритм перестановки, алгоритм скремблеров, алгоритм замены по таблице, матричный метод преобразования и алгоритм Винжера.
2. Изучить режимы использования блочных шифров (ECB, CBC, CFB и OFB).
3. Изучить способы объединения блочных шифров (многократное шифрование, сеть Фейстела).
4. Реализовать систему в соответствии с вариантами, указанными в таблице и заданием:
5. Разработать собственный алгоритм который реализует указанный в варианте:
    - режим использования блочного шифра;
    - работает с указанной длиной блока;
    - позволяет оценивать скорость шифрования/дешифрования.
    -

3. Замена по таблице CBC 1 байт Число столбцов 5

Ход работы:

CBCTable.h:

```
#pragma once
#include <vector>
#include <bitset>
#include <string>
#include <iostream>
#include <fstream>
class CBCTable
{
private:
	std::vector<std::vector<std::string>> translationTable;
	std::string charToBinary(char character);
	char binaryToChar(const std::string& binaryString);
	std::string applyXor(const std::string& binaryString1, const std::string&
binaryString2);
	std::string encryptThroughTable(std::string);
	std::string decryptThroughTable(std::string);
public:
	CBCTable();
	void encryptStandard(std::string path);
	void decryptStandard(std::string path);
	void encryptDouble(std::string path); //двойное шифрование
	void decryptDouble(std::string path);
};
```

CBCTable.cpp:

```
#include "CBCTable.h"
std::string CBCTable::charToBinary(char character) {
std::bitset<8> bits(character);
return bits.to_string();
}
char CBCTable::binaryToChar(const std::string& binaryString) {
std::bitset<8> bits(binaryString);
return static_cast<char>(bits.to_ulong());
}
std::string CBCTable::applyXor(const std::string& binaryString1, const std::string&
binaryString2) {
std::bitset<8> bits1(binaryString1);
std::bitset<8> bits2(binaryString2);
```

```
std::bitset<8> result = bits1 ^ bits2;
return result.to_string();
}
std::string CBCTable::encryptThroughTable(std::string ch) {
for (int i = 0; i < translationTable.size(); i++)
for (int j = 0; j < translationTable[i].size(); j++)
if (translationTable[i][j] == ch) {
if (i == 50)
return translationTable[0][j];
else
return translationTable[i + 1][j];
}
return "*";
}
std::string CBCTable::decryptThroughTable(std::string ch) {
for (int i = 0; i < translationTable.size(); i++)
for (int j = 0; j < translationTable[i].size(); j++)
if (translationTable[i][j] == ch) {
if (i == 0)
return translationTable[50][j];
else
return translationTable[i - 1][j];
}
return "*";
}

bool containsCharacter(const std::string& str, char character) {
return str.find(character) != std::string::npos;
}

CBCTable::CBCTable() {
translationTable = std::vector<std::vector<std::string>>(51,
std::vector<std::string>(5));
std::string key = "GIZMO";

for (int i = 0; i < translationTable[0].size(); i++)
translationTable[0][i] = charToBinary(key[i]);

for (int i = 1; i < translationTable.size(); i++)
for (int j = 0; j < translationTable[i].size(); j++) {
char character = binaryToChar("00000000");
while (containsCharacter(key, character))
character++;
translationTable[i][j] = charToBinary(character);
key += character;
}
}

void CBCTable::encryptStandard(std::string path) {
std::ifstream fin(path);
std::ofstream fout("encrypted1.txt");
std::string initializationVector = charToBinary('A');
char curCharacter;

while (fin.get(curCharacter)) {
```

```
std::string res = charToBinary(curCharacter);
res = applyXor(res, initializationVector);
res = encryptThroughTable(res);
fout << binaryToChar(res);
initializationVector = res;
}
fin.close();
fout.close();
}

void CBCTable::decryptStandard(std::string path) {
std::ifstream fin(path);
std::ofstream fout("decrypted1.txt");
std::string initializationVector = charToBinary('A');
char curCharacter;

while (fin.get(curCharacter)) {
std::string res = charToBinary(curCharacter);
res = decryptThroughTable(res);
res = applyXor(res, initializationVector);
fout << binaryToChar(res);
initializationVector = charToBinary(curCharacter);
}
fin.close();
fout.close();
}

void CBCTable::encryptDouble(std::string path) {
std::ifstream fin(path);
std::ofstream fout("encrypted2.txt");
std::string initializationVector = charToBinary('A');
char curCharacter;

while (fin.get(curCharacter)) {
std::string res = charToBinary(curCharacter);
res = applyXor(res, initializationVector);
res = encryptThroughTable(res);
res = encryptThroughTable(res);
fout << binaryToChar(res);
initializationVector = res;
}
fin.close();
fout.close();
}

void CBCTable::decryptDouble(std::string path) {
std::ifstream fin(path);
std::ofstream fout("decrypted2.txt");
std::string initializationVector = charToBinary('A');
char curCharacter;

while (fin.get(curCharacter)) {
std::string res = charToBinary(curCharacter);
res = decryptThroughTable(res);
res = decryptThroughTable(res);
```

```
res = applyXor(res, initializationVector);
fout << binaryToChar(res);
initializationVector = charToBinary(curCharacter);
}
fin.close();
fout.close();
}
```

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