Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №2

по курсу «ИТОКБ»

на тему: «Простейшие криптографические преобразования»

Вариант 4

|  |  |
| --- | --- |
| Выполнил магистрант группы 025941: | Колесников В.Г. |
| Проверил: | Боброва Т.С. |

МИНСК

2021

**Задание 1:** Реализовать в виде программы шифр (зашифрование и расшифрование) перестановки, использующий простые (прямоугольные) таблицы. Язык исходного текста — английский.

Код программы приведен в конце лабораторной работы.

Результаты работы программы зашифрования и расшифрования приведены на рисунках 1.1 и 1.2.

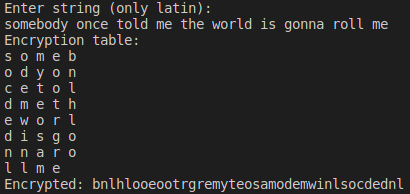


Рисунок 1.1 — Результат зашифрования текста шифром перестановки

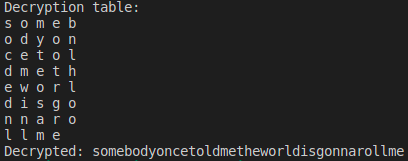


Рисунок 1.2 — Результат расшифрования текста, зашифрованного шифром перестановки

**Задание 2:** Реализовать в виде программы атаку полным перебором ключа, используя для оценки правильности выбора ключа визуальный метод или исходный текст для автоматического сравнения результата дешифрования.

Используемый шифр в качестве ключа задействует размер таблицы. Таким образом, чтобы узнать ключ, достаточно подобрать точное количество столбцов. При шифровании текста таким способом важно учитывать его размер, т. к. если количество символов больше, чем колонок в столбце, шифрование не удастся. Зная алгоритм шифрования, в данном случае расшифровка не составит труда при использовании вычислительных средств.

Зашифрованный текст с ключом в 100 колонок представлен на рисунке 2.1. Расшифрованный текст и таблица дешивровки представлены на рисунке 2.2.

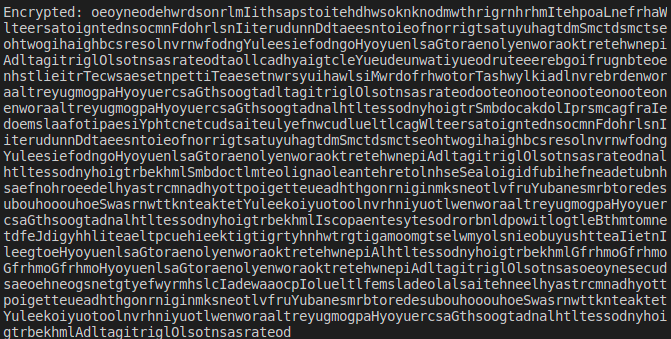


Рисунок 2.1 — Зашифрованный текст

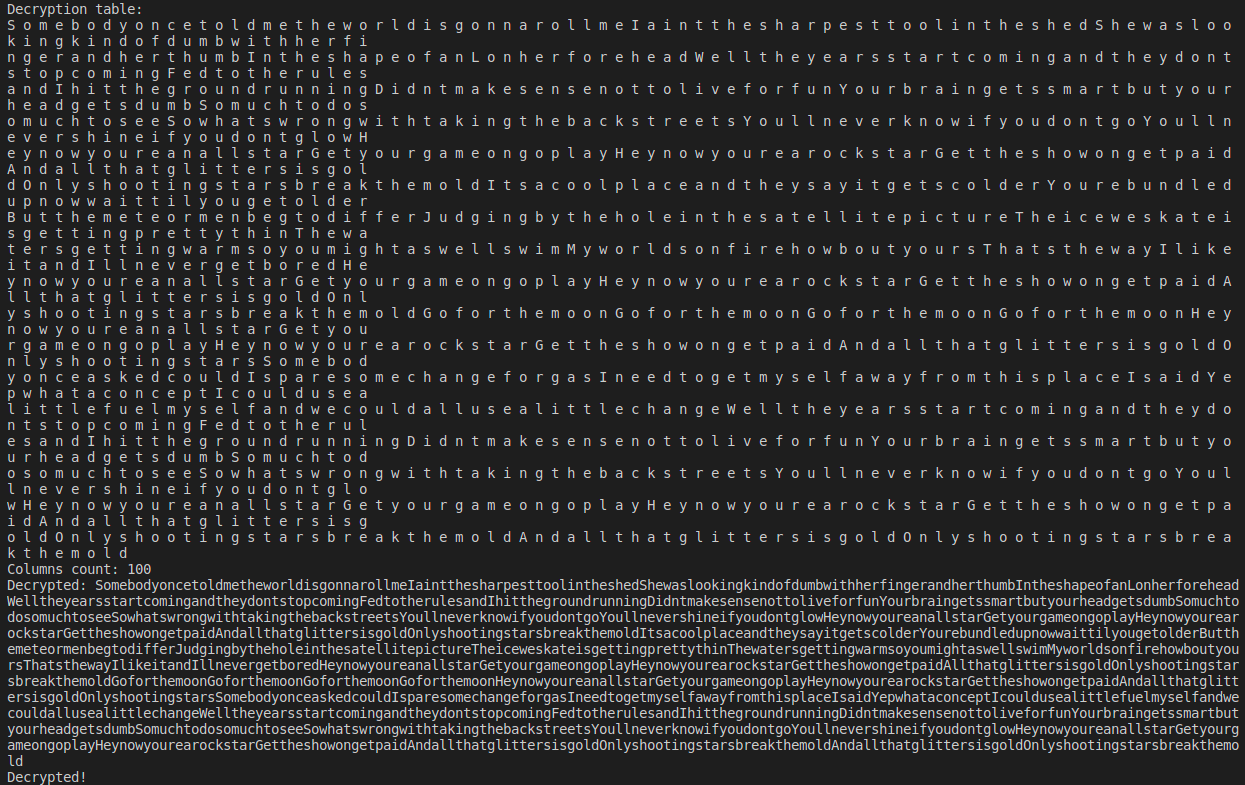


Рисунок 2.2 — Таблица дешифровки и расшифрованный текст

**Задание 3:** Дать практические рекомендации по выбору пароля исходя из предположений б алфавите пароля, ценности информации, доступ к которой защищается с помощью этого пароля, производительности вычислительного средства атакующего и времени атаки.

Время подбораэкспоненциально зависит от мощности алфавита и длины подбираемого пароля. Дольше всего будет подбираться пароль, состоящий полностью из последней буквы в алфавите (Z), быстрее всего – из первой (a).

Даже в условиях сильно ограниченного времени, 121 секунды должно хватить на подбор пароля по модели «скринсейвера», когда оператор машины временно отсутствует. Так что пароль рекомендуется делать не менее 6-ти символов.

Если в систему имеется доступ извне и нет ограничений по вводу паролей, нежелательно делать пароль короче, чем 8 символов.

Если система обладает высокой ценностью и пароль на ней меняется редко, желательно использовать пароли от 9 и более символов.

Существует несколько видов атак, которые упрощают задачу подбора пароля. Для этого во многих IT-компаниях введен стандарт — от 15 символов, состоящих из букв верхнего и нижнего регистра, цифр, а также нестандартных для алфавита знаков и специальных символов. Кроме того, рекомендуется менять пароль не реже чем через каждые три месяца для исключения его компрометации.

Для улучшенной защиты данных также могут использоваться такие методы, как двухфакторная аутентификация и одноразовые пароли.

**Код программы:**

#include <iostream>

#include <time.h>

const int createPassword(const wchar\_t \*alphabet, const int lettersCount)

{

int symbCount;

wprintf(L"Enter symbols count: ");

wscanf(L"%d", &symbCount);

wchar\_t \*pass = new wchar\_t[symbCount + 1];

for (int i = 0; i < symbCount; i++)

{

const int letterNumber = rand() % lettersCount;

// wprintf(L"Next number: %d\n", letterNumber);

pass[i] = alphabet[letterNumber];

}

pass[symbCount] = L'\0';

wprintf(L"Generated password: %ls\n", pass);

delete pass;

return 0;

}

const int createDistribution(const wchar\_t \*alphabet, const int lettersCount)

{

int \*lettersDistribution = new int[lettersCount];

for (int i = 0; i < lettersCount \* 10000; i++)

{

const int letterNumber = rand() % lettersCount;

lettersDistribution[letterNumber]++;

}

for (int i = 0; i < lettersCount; i++)

{

wprintf(L"%lc: %ld\n", alphabet[i], lettersDistribution[i]);

}

delete lettersDistribution;

return 0;

}

const int calcAvgGuessTime(const wchar\_t \*alphabet, const int lettersCount)

{

const int countOfTries = 5;

const int symbolCounts[countOfTries] = {2, 3, 4, 5, 6};

for (int symbCountNumber = 0; symbCountNumber < countOfTries; symbCountNumber++)

{

const long symbCount = symbolCounts[symbCountNumber];

wprintf(L"Symbols count: %ld\n", symbCount);

for (int tryNumb = 1; tryNumb < countOfTries + 1; tryNumb++)

{

// Password generation here

wchar\_t \*pass = new wchar\_t[symbCount + 1];

pass[symbCount] = L'\0';

for (int i = 0; i < symbCount; i++)

{

const int letterNumber = rand() % lettersCount;

pass[i] = alphabet[letterNumber];

}

wprintf(L"Password: %ls. ", pass);

// Brute force

time\_t start = time(0);

int \*bfSymbNumbers = new int[symbCount];

for (int i = 0; i < symbCount; i++)

{

bfSymbNumbers[i] = 0;

}

int bfSymbNumbersSum = 0;

bool increaseNext = false;

bool isPassBruteForced = false;

do

{

for (int i = 0; i < symbCount; i++)

{

if (increaseNext)

{

bfSymbNumbers[i]++;

}

if (bfSymbNumbers[i] > lettersCount - 1)

{

bfSymbNumbers[i] = 0;

increaseNext = true;

}

else if (i == 0)

{

bfSymbNumbers[i]++;

}

else

{

increaseNext = false;

}

}

int similarCount = 0;

for (int i = 0; i < symbCount; i++)

{

if (alphabet[bfSymbNumbers[i]] == pass[i])

{

similarCount++;

}

}

if (similarCount == symbCount)

{

break;

}

bfSymbNumbersSum++;

} while (true);

time\_t end = time(0);

// Time calculation

const long int startInt = static\_cast<long int>(start);

const long int endInt = static\_cast<long int>(end);

const long int elapsed = endInt - startInt;

wprintf(L"Guessing time: %lds\n", elapsed);

delete pass;

delete bfSymbNumbers;

}

}

return 0;

}

int main(int argc, char \*\*argv)

{

setlocale(LC\_ALL, "");

srand(time(NULL));

const wchar\_t \*alphabet = L"абвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

const int lettersCount = wcslen(alphabet);

// а

// - 1

createPassword(alphabet, lettersCount);

// - 2

createDistribution(alphabet, lettersCount);

// - 3;

calcAvgGuessTime(alphabet, lettersCount);

return 0;

}