Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу «Защита информации в сети Internet»

на тему «разработка программы шифрования/дешифрирования двоичных файлов с использованием перестановочного шифра.»

Вариант 2

Выполнили  
студенты группы 18ВВ1:

Коротов В.О.

Максимова Д.Р.

Приняли:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

**2020**

**Цель работы:** разработать программу для шифрования/дешифрирования двоичных файлов с использованием перестановочного шифра.

**Задание:** Разработать программу, выполняющую шифрование и расшифровывание произвольного двоичного файла с использованием перестановочного шифра используя в качестве ключа последовательность, соответствующую номеру варианта. Выполнить проверку путем двоичного сравнения исходного файла и фала, полученного после расшифровывания.

|  |  |
| --- | --- |
| Номер варианта | Последовательность перестановки |
| 2 | 2 28 13 15 20 12 5 24 25 27 1 3 14 9 8 21 22 4 26 23 19 30 18 6 11 17 31 16 10 7 29 0 |

**Описание работы программы:** имеется исходный файл some.exe, который шифруется по ключу. Результат шифрования записывается в файл encrypted.txt. Далее этот файл расшифровывается и записывается в файл decrypted.txt. После этого сравниваем файл с исходным some.exe и расшифрованный файл decrypted.txt. Результат сравнения выводим в консоль.

Результат работы программы:



Листинг:

#include "pch.h"

#include <iostream>

#include <bitset>

using namespace std;

const size\_t BUFFER\_SIZE = 1;

const size\_t KEY\_SIZE = 32;

const size\_t BITS\_IN\_BYTE = 8;

const int key\_to\_encrypt[KEY\_SIZE] = { 2, 28, 13, 15, 20, 12, 5, 24, 25, 27, 1,

3, 14, 9, 8, 21, 22, 4, 26, 23,

19, 30, 18, 6, 11, 17, 31, 16, 10, 7, 29, 0 };

typedef int bytes\_block;

enum TranspositionCipherMode {

ENCRYPT,

DECRYPT

};

FILE\* open\_file(const char\* filename, const char\* mode)

{

FILE \*file = fopen(filename, mode);

if (!file)

{

printf("Can't open file %s!", filename);

exit(1);

}

return file;

}

void calculate\_key(int\* key, TranspositionCipherMode mode)

{

if (mode == ENCRYPT)

{

for (int i = 0; i < KEY\_SIZE; i++)

{

key[i] = key\_to\_encrypt[i];

}

}

else if (mode == DECRYPT)

{

for (int i = 0; i < KEY\_SIZE; i++)

{

int move\_to = key\_to\_encrypt[i];

key[key\_to\_encrypt[move\_to]] = key\_to\_encrypt[i];

}

}

}

bytes\_block\* transposition\_cipher\_str(const bytes\_block\* buffer, const int key[KEY\_SIZE], size\_t readed\_bytes)

{

bytes\_block \*buffer\_after\_transposition = (bytes\_block\*)malloc(BUFFER\_SIZE \* sizeof(bytes\_block));

bitset<KEY\_SIZE> buffer\_bitset(\*buffer);

bitset<KEY\_SIZE> buffer\_after\_bitset;

for (int i = 0; i < KEY\_SIZE; i++)

{

buffer\_after\_bitset[i] = buffer\_bitset[key[i]];

}

\*buffer\_after\_transposition = (bytes\_block)buffer\_after\_bitset.to\_ulong();

return buffer\_after\_transposition;

}

void transposition\_cipher(const char\* source\_fname, const char\* output\_fname, TranspositionCipherMode mode)

{

FILE \*fsource = open\_file(source\_fname, "rb");

FILE \*foutput = open\_file(output\_fname, "wb");

int \*key = new int[KEY\_SIZE];

calculate\_key(key, mode);

bytes\_block\* buffer = (bytes\_block\*)malloc(BUFFER\_SIZE \* sizeof(bytes\_block));

bytes\_block\* buffer\_after\_transposition = NULL;

while (!feof(fsource))

{

size\_t element\_count = fread\_s(buffer, sizeof(bytes\_block), sizeof(bytes\_block), BUFFER\_SIZE, fsource);

if (element\_count == 0) break;

buffer\_after\_transposition = transposition\_cipher\_str(buffer, key, element\_count);

fwrite(buffer\_after\_transposition, sizeof(bytes\_block), BUFFER\_SIZE, foutput);

free(buffer\_after\_transposition);

}

free(buffer);

free(key);

fclose(fsource);

fclose(foutput);

}

bool file\_equals(const char\* file1\_name, const char\* file2\_name)

{

bool files\_are\_equal = true;

FILE \*file1 = open\_file(file1\_name, "rb");

FILE \*file2 = open\_file(file2\_name, "rb");

int c1, c2;

while (!feof(file1) && !feof(file2))

{

c1 = fgetc(file1);

c2 = fgetc(file2);

if (c1 == EOF || c2 == EOF) break;

if (c1 != c2)

{

files\_are\_equal = false;

break;

}

}

fclose(file1);

fclose(file2);

return files\_are\_equal;

}

int main()

{

const char \*source\_fname = "some.exe";

const char \*encrypted\_fname = "encrypted.txt";

const char \*decrypted\_fname = "decrypted.txt";

transposition\_cipher(source\_fname, encrypted\_fname, ENCRYPT);

transposition\_cipher(encrypted\_fname, decrypted\_fname, DECRYPT);

if (file\_equals(source\_fname, decrypted\_fname))

{

printf("Files are equal!\n");

}

else

{

printf("Files are not equal!\n");

}

return 0;

}

**Вывод:** разработали программу шифрования/дешифрирования двоичных файлов с использованием перестановочного шифра.