Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

—

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**Кодирование и упаковка данных**

по дисциплине «Основы Информационной Безопасности»

Выполнил

студент гр. 5131001/30002 Костенников А.В.

<*подпись*>

Руководитель

программист Вагисаров В. Б.

<*подпись*>

Санкт-Петербург

2024г.

СОДЕРЖАНИЕ

[ЦЕЛЬ РАБОТЫ 3](#_Toc1605669050)

[ХОД РАБОТЫ 4](#_Toc1993726051)

[ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ 10](#_Toc1053662956)

[ВЫВОД 12](#_Toc1470412282)

[ПРИЛОЖЕНИЕ А 13](#_Toc1275815085)

ЦЕЛЬ РАБОТЫ

Приобретение навыков по защите информации с помощью методов кодирования, получение прикладных знаний в области исследования и реализации алгоритмов упаковки данных.

ХОД РАБОТЫ

1. **Разработка утилит**

Были разработаны утилиты для реализации кодирования и декодирования по алгоритмам Фано и LZW (код разработанной программы представлен в приложении А).

1. **Текст для тестирования программы**

Был создан и помещен в директорию с программой текстовый документ, состоящий из 164245 символов a, b, c, d (рисунок 1).

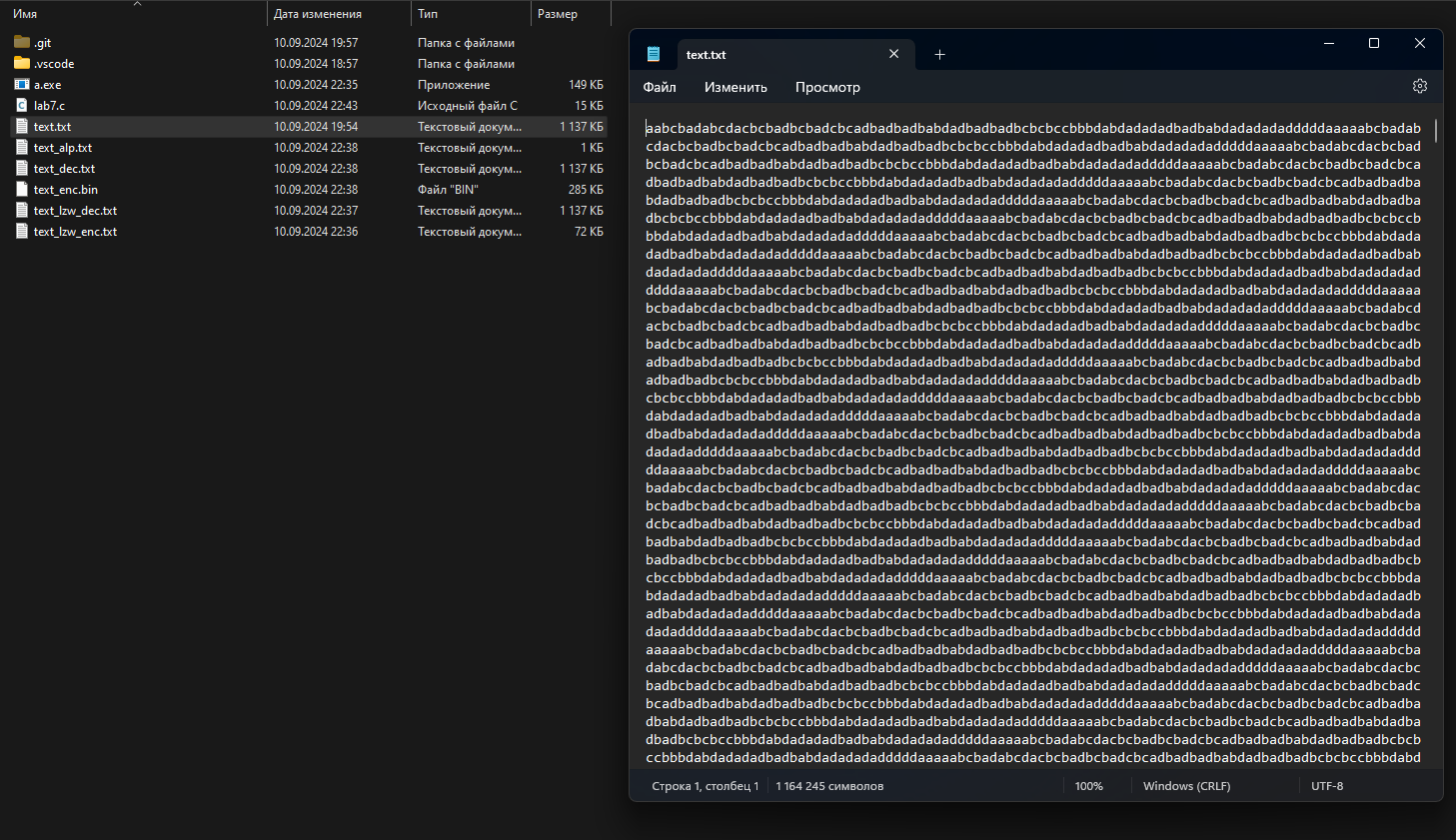


Рисунок 1 – текстовый файл для тестирования программы

1. **Кодирование и декодирование текстового файла**

Текстовый файл был закодирован сначала алгоритмом Фано, затем алгоритмом LZW.

Алгоритм Фано справился за 0.058 секунд, достигнув коэффициента сжатия 4 (рисунок 2).

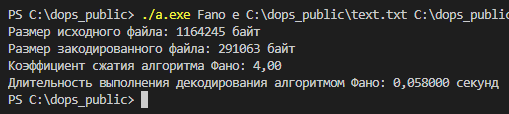


Рисунок 2 – кодирование алгоритмом Фано

Алгоритм LZW справился за 35.716 секунд, достигнув коэффициента сжатия 15.84 (рисунок 3).

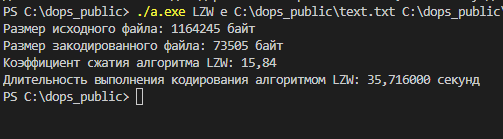


Рисунок 3 – кодирование алгоритмом LZW

После чего соответствующие файлы с закодированным текстом были декодированы.

Алгоритм Фано справился за 0.14 секунд (рисунок 4).

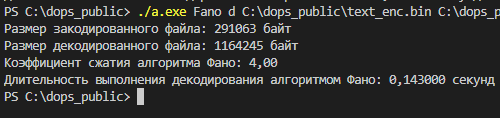


Рисунок 4 – декодирование алгоритмом Фано

Алгоритм LZW справился за 0.018 секунд (рисунок 5).

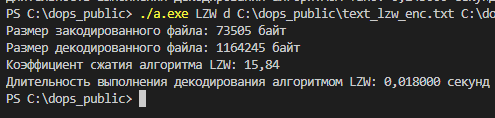


Рисунок 5 – декодирование алгоритмом LZW

Полученные декодированные файлы полностью совпадают с исходным (рисунок 6).

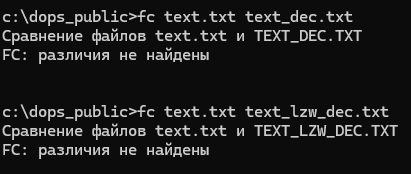


Рисунок 6 – анализ декодированных файлов

Закодируем файл через алгоритм сначала LZW, затем Фано (рисунок 7).

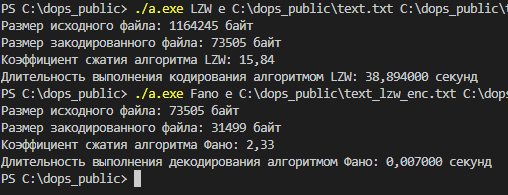


Рисунок 7 – кодирование двумя алгоритмами

Декодируем полученный файл сначала алгоритмом Фано, затем LZW (рисунок 8).

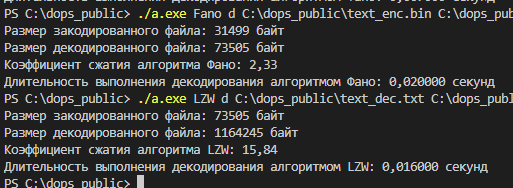


Рисунок 8 – декодирование двумя алгоритмами

Полученный файл совпадает с исходным (рисунок 9).

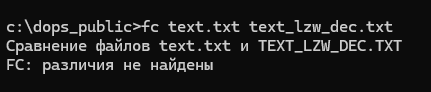


Рисунок 9 – анализ декодированного файла после комбинированного кодирования

1. **Кодирование и декодирование бинарного файла (файла в формате word)**

Выберем числа и . Методом Миллера докажем, что число составное (формула 7), и покажем, что число простое (формула 9).

Число – нечетное, тогда представим , где t – нечетно (формула 6).

После проверки мы получили, что является составным.

Аналогично представим , где t – нечетно для (формула 8)

Получили, что N – простое согласно алгоритму, но вероятность этого 25%, т.к. определение выполнялось по методу Миллера.

1. **Изучение RSA шифрования (Пункт 5, 6, 7, 8)**

Выберем числа p = 3433 и q = 2621.

Вычислим число n (формула 10).

Вычислим порядок группы ф(n) (формула 11).

Выберем показатель шифрования e = 7.

Вычислим показатель d (формула 12).

Получили d = 6422743

Возьмем текст «Hello, vshk!», переведем его в ASCII-код: «72 101 108 108 111 44 32 118 115 104 107 33». Так получим числа x = 72, 101, 108, и т.д. Зашифруем их (формула 13).

Попробуем расшифровать полученный текст y (формула 14).

Можем сделать вывод, что полученный после расшифрования текст совпадает с исходным.

Подпишем x цифровой подписью s (формула 15)

Вычислим обратное преобразование x (формула 16).

Получаем, что исходный текст совпадает с полученным.

Смоделируем процесс установления сеансового ключа.

Возьмем a = 123, x = 7777, y = 33333.

Для пользователя A получаем промежуточное сообщение A (формула 17) и конечное сообщение (формула 19).

Для пользователя B получаем промежуточное сообщение B (формула 18) и конечное сообщение (формула 20).

Получаем, что конечные сообщения пользователей A и B совпадают, значит равенство (формула 21) верно.

1. **Алгоритм ранцевой криптосистемы Меркля-Хеллмана (Пункт 9)**

Простая реализация утилиты шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана представлена в приложении А (листинг 1).

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое вычет? На чем основан алгоритм шифрования Цезаря?

Вычет - это результат деления одного целого числа на другое с остатком. В криптографии вычеты используются для определения результата шифрования и расшифрования сообщений.

Алгоритм шифрования Цезаря основан на принципе сдвига символов в алфавите на определенное количество позиций. Ключевая идея заключается в том, что каждая буква сообщения сдвигается на фиксированное количество позиций в алфавите. Например, при сдвиге на 3 позиции буква "A" становится "D", "B" становится "E" и так далее. Этот сдвиг задается ключом шифра.

Таким образом, алгоритм Цезаря использует вычеты для определения нового положения символов в алфавите после сдвига, а ключ шифра определяет, на сколько позиций происходит этот сдвиг.

1. Каковы особенности чисел Кармайкла?

Числами Кармайкла называются составные числа N такие, что они удовлетворяют условию малой теоремы Ферма по определению простых чисел «если число N простое, то для любого целого a, не делящегося на N, выполняется сравнение »

Таким образом: Числа Кармайкла - это составные натуральные числа, которые обладают свойством псевдопростоты по отношению ко всем основаниям основаниям системы счисления. Это означает, что число Кармайкла может быть простым по виду для некоторых оснований системы счисления, но на самом деле оно не является простым.

1. Перечислите основные свойства мультипликативной группы кольца вычетов по модулю pq.

Мультипликативная группа этого кольца является абелевой и циклической, состоит из ненулевых чисел, меньших n и взаимно простых с n. Остатки от деления образующей группы на p и q равны Fp\* и Fq\*.

Порядок группы равен значению функции Эйлера от n.

Любой элемент кольца может быть единственным образом представлен в виде a(modp) и a(modq) и обратно.

Единичным элементом в этой группе является элемент, сравнимый с 1 по модулю q и сравнимый с 0 по модулю p.

1. Почему порядок группы (Z/nZ)\* должен иметь большой простой делитель?

Порядок группы (𝑍/𝑛𝑍)\*, также известной как мультипликативная группа вычетов по модулю 𝑛n, представляет собой количество элементов в этой группе. Этот порядок играет важную роль в различных алгоритмах криптографии, основанных на теории чисел, таких как RSA.

В криптографических алгоритмах, таких как RSA, сложность взлома алгоритма напрямую связана с факторизацией больших чисел. Если порядок группы (𝑍/𝑛𝑍)∗ имеет большой простой делитель, это делает задачу факторизации числа n более сложной, что повышает безопасность алгоритма. Большой простой делитель порядка группы (Z/nZ)∗ также повышает стойкость алгоритма к различным атакам, включая атаки перебором и атаки на основе структуры группы.

1. Опишите алгоритм расчета кодов символов при декодировании шифрограмм согласно алгоритму Меркля-Хеллмана.

Будем считать, что мы имеем зашифрованное сообщение x, числа m и n, шифрограмму c.

Необходимо найти число , что .

После каждое значение шифрограмм c умножить на по модулю m.

Мы получим сумму «весов» для битов двоичной записи символа x.

Так, если мы имеем закрытый ключ (сверхвозрастающую последовательность), мы сможем однозначно расшифровать символ, основываясь на преобразованной шифрограмме, идя от большего «веса» к меньшему, где соответствующему «весу» присваиваем значение «1» для бита двоичной записи.

Переводим двоичную запись в текст x.

ВЫВОД

В работе были рассмотрены математические основы криптографии, такие как некоторые распространенные алгоритмы и методы шифрования: шифр Цезаря, алгоритм Евклида, метод Миллера-Рабина, алгоритмы шифрования RSA и ранцевая криптосистема Меркля-Хеллмана. При этом последний метод шифрования был реализован в виде программы на языке программирования C.

ПРИЛОЖЕНИЕ А

Листинг 1 - исходный код программы на языке С

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

int euklid(int n\_max, int n\_min){

int temp;

if(n\_min == 0){

return n\_max;

}

return euklid(n\_min, n\_max%n\_min);

}

void func(char symbol, int binary\_seq[8]){

int temp = (int) symbol;

int cur = 7;

while(temp > 0 && cur >= 0){

binary\_seq[cur] = (temp % 2) ? 1 : 0;

cur--;

temp/=2;

}

if(cur > 0){

while(cur >= 0){

binary\_seq[cur] = 0;

cur--;

}

}

}

int unfunc(int unbinary\_seq[8]){

int temp = 0;

for(int i = 7; i >= 0; i--){

temp += unbinary\_seq[i] \* (int) pow(2.0, (float) (7 - i));

}

return temp;

}

int find\_inverse(int n, int m) {

for (int n1 = 1; n1 < m; n1++) {

if ((n \* n1) % m == 1) {

return n1;

}

}

return -1;

}

int main(void){

int min\_m = 0;

char message[] = "Hello, vshk!";

int len\_of\_mess = (sizeof(message)/ sizeof(char))-1;

int len\_of\_seq = 8;

// Создание сверхвозрастающей последовательности

int \*privatekey = NULL;

privatekey = (int\*) malloc(len\_of\_seq \* sizeof(int));

srand(time(NULL));

privatekey[0] = 1 + rand()%10;

min\_m = privatekey[0];

for(int i = 1; i < len\_of\_seq; i++){

privatekey[i] = min\_m + 1 + rand()%10;

min\_m += privatekey[i];

}

// Определение m и n

int m = min\_m + 1;

int n = 0;

for(int i = 2; i < m; i++){

if(euklid(m, i) == 1){

n = i;

break;

}

}

// Поиск обратного элемента

int n1 = find\_inverse(n, m);

// Создание открытого ключа

int \*publickey = NULL;

publickey = (int\*) malloc(len\_of\_mess \* sizeof(int));

for(int i = 0; i < len\_of\_seq; i++){

publickey[i] = (privatekey[i]\*n)%m;

}

// Создание зашифрованного сообщения

int \*key = NULL;

key = (int\*) malloc(len\_of\_mess \* sizeof(int));

for(int i = 0; i < len\_of\_mess; i++){

int binary\_seq[8] = {0};

key[i] = 0;

func(message[i], binary\_seq);

for(int j = 0; j < len\_of\_seq; j++){

key[i] += binary\_seq[j] \* publickey[j];

}

}

for(int i = 0; i < len\_of\_mess; i++){

printf("%d ", key[i]);

}

printf("\n");

// Расшифрование сообщения

char \*res = NULL;

res = (char\*) malloc(len\_of\_mess \* sizeof(char));

for(int i = 0; i < len\_of\_mess; i++){

key[i] = (key[i]\*n1)%m;

int unbinary\_seq[8] = {0};

for(int j = 7; j >= 0; j--){

if (key[i] >= privatekey[j]){

unbinary\_seq[j] = 1;

key[i] -= privatekey[j];

}

}

res[i] = (char) unfunc(unbinary\_seq);

printf("%c", (char) unfunc(unbinary\_seq));

}

free(privatekey);

free(key);

free(res);

return 0;

}