Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине «Криптографические методы защиты информации»

Тема: «Криптографические хэш-функции»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Карагодин Д.Л.

**Проверил:**

Хацкевич А.С.

Брест 2023

**Цель работы:** Изучить существующие алгоритмы вычисления дайджестов сообщений и написать программу, реализующую заданный алгоритм хэширования.

**Ход работы:**

**Вариант 6 (**Алгоритм SHA–1**)**

**Задание 1.**

Реализовать приложение, позволяющее вычислять значение хэш-функции, заданной в варианте:

* + - текст сообщения должен считываться из файла;
    - полученное значение хэш-функции должно представляться в шестнадцатеричном виде и сохраняться в файл;
    - при работе программы должна быть возможность просмотра и изменения считанного из файла сообщения и вычисленного значения хэш-функции.
    - Исследовать лавинный эффект.

**Задание 2.**

Вычислить значение хеш-функции с помощью технологии OpenSSL.

**Задание 1.**

**Код программы:**

import struct

import hashlib

import matplotlib.pyplot as plt

import os

def calculate\_hash(file\_path):

    with open(file\_path,encoding="UTF-8",mode='r') as file:

        data = file.read()

        hash\_object = sha1(data)

        return hash\_object

def save\_hash(file\_path, hash\_value):

    with open(file\_path,encoding="UTF-8",mode= 'w') as file:

        file.write(hash\_value)

def read\_message(file\_path):

    with open(file\_path,encoding="UTF-8",mode= 'r') as file:

        return file.read()

def save\_message(file\_path, message):

    with open(file\_path,encoding="UTF-8",mode= 'w') as file:

        file.write(message)

def left\_rotate(n, b):

    return ((n << b) | (n >> (32 - b))) & 0xffffffff

def sha1(message):

    h0,h1,h2,h3,h4 = 0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476, 0xC3D2E1F0

    message = bytearray(message, 'utf-8')

    length = (8 \* len(message)).to\_bytes(8, byteorder='big')

    message.append(0x80)

    while len(message) % 64 != 56:

        message.append(0x00)

    message += length

    for i in range(0, len(message), 64):

        block = message[i:i+64]

        words = struct.unpack('>16L', block)

        w = list(words) + [0] \* 64

        for j in range(16, 80):

            w[j] = left\_rotate((w[j-3] ^ w[j-8] ^ w[j-14] ^ w[j-16]), 1)

        a,b,c,d,e = h0,h1,h2,h3,h4

        for j in range(80):

            if j <= 19:

                f = (b & c) | ((~b) & d)

                k = 0x5A827999

            elif j <= 39:

                f = b ^ c ^ d

                k = 0x6ED9EBA1

            elif j <= 59:

                f = (b & c) | (b & d) | (c & d)

                k = 0x8F1BBCDC

            else:

                f = b ^ c ^ d

                k = 0xCA62C1D6

            temp = (left\_rotate(a, 5) + f + e + k + w[j]) & 0xffffffff

            e,d,c,b,a = d,c,left\_rotate(b, 30),a,temp

        h0 = (h0 + a) & 0xffffffff

        h1 = (h1 + b) & 0xffffffff

        h2 = (h2 + c) & 0xffffffff

        h3 = (h3 + d) & 0xffffffff

        h4 = (h4 + e) & 0xffffffff

    hash\_value = '{:08x}{:08x}{:08x}{:08x}{:08x}'.format(h0, h1, h2, h3, h4)

    return hash\_value

def count\_changed\_bits(hash1, hash2):

    hash1,hash2=str(hash1),str(hash2)

    count = 0

    for byte1, byte2 in zip(hash1, hash2):

        diff = int(byte1) ^ int(byte2)

        count += bin(diff).count('1')

    return count

def compute\_hash(message):

    hash\_values = []

    hash\_obj = hashlib.sha1()

    hash\_obj.update(message.encode('utf-8'))

    hash\_values.append(int.from\_bytes(hash\_obj.digest(), 'big'))

    for pos in range(len(message) \* 8):

        modified\_message = message[:pos // 8] + \

                           chr(ord(message[pos // 8]) ^ (1 << (7 - pos % 8))) + \

                           message[pos // 8 + 1:]

        hash\_obj = hashlib.sha1()

        hash\_obj.update(modified\_message.encode('utf-8'))

        hash\_values.append(int.from\_bytes(hash\_obj.digest(), 'big'))

    return hash\_values

def plot\_bit\_changes(hash\_values):

    rounds = range(len(hash\_values)-1)

    bit\_changes = [count\_changed\_bits(hash\_values[i], hash\_values[i+1]) for i in range(len(hash\_values)-1)]

    plt.plot(rounds, bit\_changes)

    plt.xlabel('Раунд вычисления хэша')

    plt.ylabel('Число измененных бит')

    plt.title('Зависимость числа измененных бит от раунда вычисления хэша')

    plt.show()

file\_path = 'message.txt'

hash\_path = 'hash.txt'

choice = ''

while choice != '5':

    print('Выберите действие:')

    print('1. Просмотреть сообщение')

    print('2. Изменить сообщение')

    print('3. Вычислить и сохранить значение хэш-функции')

    print('4. Лавинный эффект')

    print('5. Выйти')

    choice = input('Введите номер действия: ')

    os.system("cls")

    if choice == '1':

        message = read\_message(file\_path)

        print('Сообщение:', message)

    elif choice == '2':

        message = input('Введите новое сообщение: ')

        save\_message(file\_path, message)

        print('Сообщение успешно изменено.')

    elif choice == '3':

        hash\_value = calculate\_hash(file\_path)

        save\_hash(hash\_path, hash\_value)

        print('Значение хэш-функции сохранено.')

    elif choice == '4':

        message = read\_message(file\_path)

        hash\_values = compute\_hash(message)

        plot\_bit\_changes(hash\_values)

        print('Изучение лавинного эффекта завершено.')

    elif choice == '5':

        print('Выход.')

    else:

        print('Некорректный выбор. Попробуйте снова.')

**Информация в файле message.txt:**

Hello

**Результат программы:**

Сообщение: Hello

Выберите действие:

1. Просмотреть сообщение

2. Изменить сообщение

3. Вычислить и сохранить значение хэш-функции

4. Лавинный эффект

5. Выйти

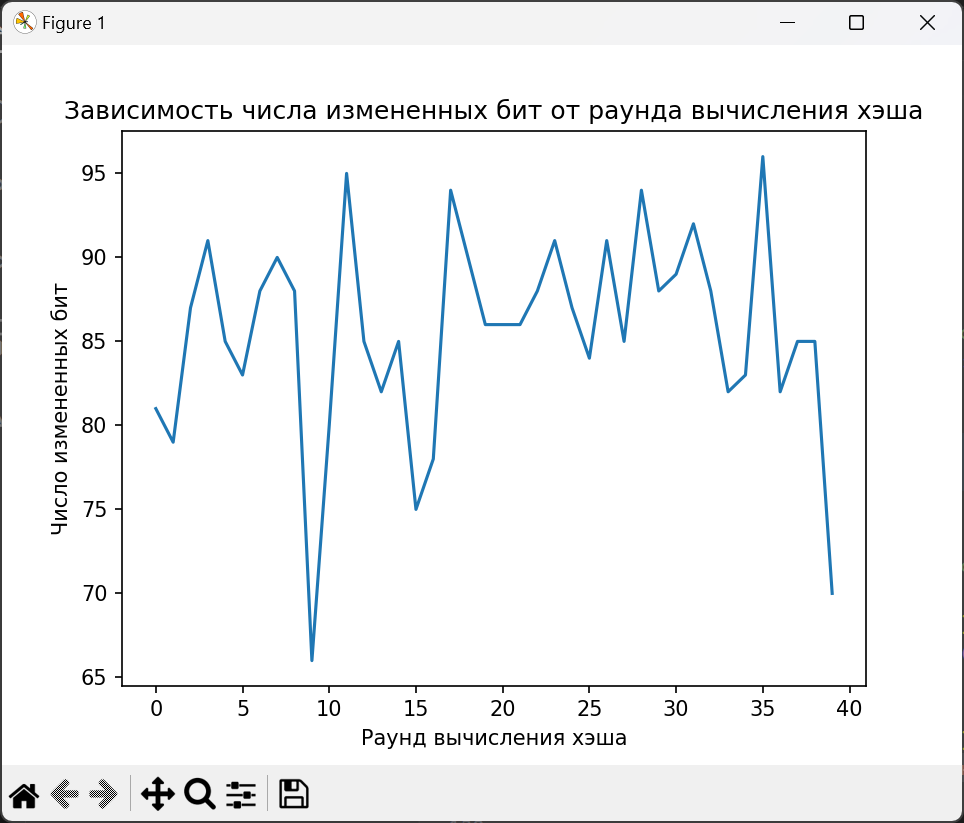
Введите номер действия: 3

Значение хэш-функции сохранено.

**Информация в файле hash.txt:**

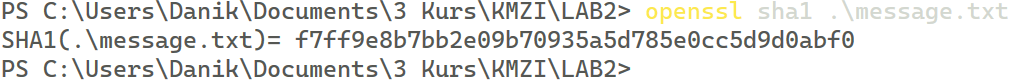
f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

**График зависимости числа измененных бит от раунда вычисления хэша:**



**Задание 2.**

**Результат программы:**

****

**Вывод:** Изучил существующие алгоритмы вычисления дайджестов сообщений и написал программу, реализующую заданный алгоритм хэширования.