Національний авіаційний університет

Факультет комп’ютерних наук та технологій

**ЗВІТ**

**по лабораторній роботі No 4**

**Однонаправлені хеш-функції.**

Дисципліна: «Криптологія»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 3 курсу. 351 групи

Архіпов Олексій Тімурович

Перевірила: к.т.н.доц.Чолишкіна О.Г.

Київ 2023

**Тема:** Однонаправлені хеш-функції.

**Ціль**: Ознайомитися з різними алгоритмами формування хеш-функцій. Вивчити їх області застосування і основні властивості. Розглянути класи атак, спрямованих на аналіз хешфункцій. Вивчити структуру сучасних методів формування хеш-функцій: SHA-1 і MD5.

Розглянути найпростіші хеш-функції.

**Завдання**:

1. Вивчити основні теоретичні положення щодо використання та формування

односпрямованих хеш-функцій.

2. Реалізувати найпростішу функцію хешування, що використовує операції XOR з застосуванням рандомізації:

a. В якості вихідного повідомлення М прийняти перші дванадцять символів з

гами «прізвищеім’я», символи набираються латиницею;

b. Розрядність хеша вибрати рівною 16-ти бітам;

c. Значення хеш-функції зберегти в файлі "hash.txt".

**Короткі теоретичні відомості**

**MD5** — 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом в 1991 році. Призначений для створення «відбитків» або «дайджестів» повідомлень довільної довжини. Прийшов на зміну MD4, що був недосконалим.

**Початковий етап підготовки**

* Вхідні дані вирівнюються так, щоб їхній розмір можна було порівняти з 448 по модулю з 512. Спочатку дописують одиничний біт (навіть якщо довжина порівняна з 448), далі необхідна кількість нульових бітів.
* Дописування 64-бітного представлення довжини даних по вирівнюванню. Якщо довжина перевищує 264−12^64 - 1, то дописують молодші біти.

**Допоміжні таблиці та функції**

Ініціалізуть 4 змінних розміром по 32 біта:

А: 01 23 45 67,

В: 89 АВ CD EF,

С: FE DC ВА 98,

D: 76 54 32 10.

Вирівнювані дані розбиваються на блоки по 32 біта, і кожен проходить 4 раунди з 16 операторів.

,

де

а, b, с, d — чотири слова буфера в певному порядку, що змінюються в процесі роботи,

g - одна з примітивних функцій F, G, Н, I,

<< 5 — циклічний зсув вліво (поворот) 32-бітового аргументу на 5 біт,

X[k] - k -е 32-битовое слово в q -м 512-бітовим блоком повідомлення.

T[i] - i -е 32-бітовое слово в матриці T,

+ - складання по модулю 2^32.



Рис.1. Елементарна операція MD5 (один крок раунду)

За допомогою MD5 перевіряли цілісність та справжність завантажених файлів – так, деякі програми поставляються разом зі значенням контрольної суми. Наприклад, пакети для інсталяції вільного ПЗ .

MD5 використовувався для хешування паролів. У системі UNIX кожен користувач має свій пароль, і його знає лише користувач. Для захисту паролів використовується хешування. Передбачалося, що отримати цей пароль можна лише повним перебором.

Існує думка, що зламати хеш MD5 неможливо, проте це неправда, існує безліч програм, що підбирають вихідне слово на основі хеша. Абсолютна більшість з них здійснює перебір за словником, проте існують такі методи як Rainbow Crack, він заснований на генеруванні безлічі хешей з набору символів, щоб по базі, що вийшла, здійснювати пошук хеша.  
Також у MD5, як у будь-якої хеш-функції, існує таке поняття як колізії - це отримання однакових хеш для різних вихідних рядків. У 1996 році Ганс Доббертін знайшов псевдоколізії в MD5, використовуючи певний буфер, що ініціалізує (ABCD). Також у 2004 році китайські дослідники Ван Сяоюнь, Фен Денгуо, Лай Сюецзя та Юй Хунбо оголосили про виявлену ними вразливість у алгоритмі, що дозволяє за невеликий час (1 годину на кластері IBM p690) знаходити колізії. Однак у 2006 році чеський дослідник Властиміл Кліма опублікував алгоритм, що дозволяє знаходити колізії на звичайному комп'ютері з будь-яким початковим вектором (A,B,C,D) за допомогою методу, який він назвав «тунелюванням».

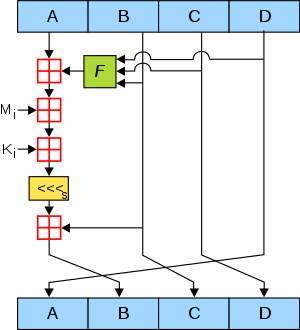


Рис.2. Блок-схема MD5

**Робота програми**

Після запуску програми відкривається вікно, в якому потрібно ввести слово для якого потрібно повернути хеш:

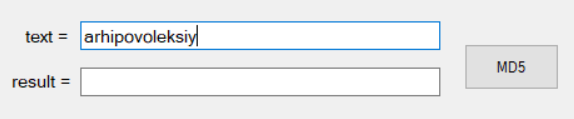


Рис.3. Вигляд вікна

Після натискання на MD5 у комірці result з’явиться хеш і він автоматично запишеться у тестовий файл:

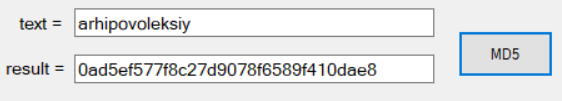


Рис.4. Результат роботи програми

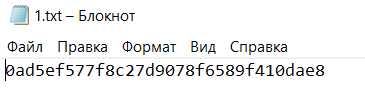


Рис.5. Результат роботи програми записаний у файл

**Висновки**

1. Ознайомився з однонаправленими хеш-функціями.
2. Реалізував програмно алгоритм MD5.
3. Реалізував запис результату програми у текстовий файл.

**Список використаної літератури**

1. Учасники проектів Вікімедіа. MD5 – Вікіпедія. *Вікіпедія*. URL: <https://uk.wikipedia.org/wiki/MD5>.
2. Алгоритм md5. *StudFiles*. URL: <https://studfile.net/preview/9468983/page:17/>.
3. Хеш-функція MD5. *Хабр*. URL: <https://habr.com/ru/sandbox/26876/> (дата звернення: 12.04.2023).

**Додаток**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace lab4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

void MD5()

{

byte[] text = Encoding.UTF8.GetBytes(tbtext.Text);

List<byte> list;

list = text.ToList();

if(text.Length % 64 != 0)

{

list.Add(0x80);

if(list.Count %64 != 0)

{

do

{

list.Add(0);

} while (list.Count % 64 != 0);

}

text = list.ToArray();

}

uint A = 0x67452301;

uint B = 0xefcdab89;

uint C = 0x98badcfe;

uint D = 0x10325476;

for(int i = 0; i < text.Length; i += 64)

{

uint[] X = new uint[16];

for(int j = 0; j < 16; j++)

{

X[j] = BitConverter.ToUInt32(text, i \* 64 + j \* 4);

}

uint A1 = A, B1 = B, C1 = C, D1 = D;

for (int j = 0; j < 64; j++)

{

A = B + ((A + G(B, C, D, j) + X[g(j)] + T(j)) << S(i));

uint t = D;

D = C; ;

C = A;

B = t;

}

A = A + A1;

B = B + B1;

C = C + C1;

D = D + D1;

}

byte[] a = BitConverter.GetBytes(A);

byte[] b = BitConverter.GetBytes(B);

byte[] c = BitConverter.GetBytes(C);

byte[] d = BitConverter.GetBytes(D);

List<byte[]> W = new List<byte[]>();

W.Add(a);

W.Add(b);

W.Add(c);

W.Add(d);

tbres.Text = "";

for(int i =0;i < 4; i++)

{

for (int j = 0; j < a.Length; j++)

{

tbres.Text += W[i][j].ToString("x2");

}

}

File.WriteAllText("E:\\Е\\Криптология\\lab4\\1.txt", tbres.Text);

}

uint T(int i)

{

uint t = (uint)(Math.Pow(2, 32) \* Math.Abs(Math.Sin(i + 1)));

return t;

}

int g(int i)

{

int g1 = 0;

if (i < 16)

{

g1 = i;

}

else if (i > 15 && i < 32)

{

g1 = (5 \* i + 1) % 16;

}

else if (i > 31 && i < 48)

{

g1 = (3 \* i + 5) % 16;

}

else if (i > 47 && i < 64)

{

g1 = (7 \* i) % 16;

}

return g1;

}

uint G(uint b,uint c,uint d, int i)

{

uint g = 0;

if(i < 16)

{

g = (b & c) | (~b & d);

}

else if(i > 15 && i < 32)

{

g = (b & d) | (c & ~d);

}

else if (i > 31 && i < 48)

{

g = b ^ c ^ d;

}

else if (i > 47 && i < 64)

{

g = c ^ (b | ~d);

}

return g;

}

int S(int i)

{

int[] s = new int[] { 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22 };

return s[i % 16];

}

private void btnMD5\_Click(object sender, EventArgs e)

{

MD5();

}

}

}