Міністерство освіти і науки України

Національний технічний університет «ХПІ»

Навчально-науковий інститут комп’ютерних наук та інформаційних технологій

Кафедра комп’ютерної інженерії та програмування

**ЗВІТ**

з лабораторної роботи

з дисципліни «СУЧАСНІ ТЕХНОЛОГІЇ БЕЗПЕЧНОГО ПРОГРАМУВАННЯ»

**«Хешування»**

Виконав: студент групи КН-М922б

\_Кліщов Б. Р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Перевірив: доцент кафедри

\_Бульба С. С \_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Харків - 2022

**Мета:** Дослідити і реалізувати механізм хешування алгоритму SHA.

**Завдання:**

Дослідити існуючі механізми гешування. Реалізувати алгоритм гешування SHA (будь-якої версії). Реализацію інших алгоритмів гешування слід омовити з викладачем.

Довести коректність роботи реалізованого алгоритму шляхом порівняння результатів з існуючими реалізаціями

**Хід роботи**

**Ключові елементи програми:**

var K = [64]uint32{

0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,

0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,

0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,

0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,

0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,

0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,

0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,

0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2,

}

const chunkSize = 64

func pad(message []byte) []byte {

L := make([]byte, 8)

binary.BigEndian.PutUint64(L, uint64(len(message)\*8))

message = append(message, 0x80)

for (len(message)+8)%64 != 0 {

message = append(message, 0x00)

}

message = append(message, L...)

return message

}

func Hash(message []byte) [32]byte {

message = pad(message)

// Initialize round constants

h0, h1, h2, h3, h4, h5, h6, h7 := uint32(0x6a09e667), uint32(0xbb67ae85), uint32(0x3c6ef372), uint32(0xa54ff53a),

uint32(0x510e527f), uint32(0x9b05688c), uint32(0x1f83d9ab), uint32(0x5be0cd19)

for chunkStart := 0; chunkStart < len(message); chunkStart += chunkSize {

var w [64]uint32

for i := 0; i\*4 < chunkSize; i++ {

w[i] = binary.BigEndian.Uint32(message[chunkStart+i\*4 : chunkStart+(i+1)\*4])

}

for i := 16; i < 64; i++ {

s0 := bits.RotateLeft32(w[i-15], -7) ^ bits.RotateLeft32(w[i-15], -18) ^ (w[i-15] >> 3)

s1 := bits.RotateLeft32(w[i-2], -17) ^ bits.RotateLeft32(w[i-2], -19) ^ (w[i-2] >> 10)

w[i] = w[i-16] + s0 + w[i-7] + s1

}

a, b, c, d, e, f, g, h := h0, h1, h2, h3, h4, h5, h6, h7

for i := 0; i < 64; i++ {

S1 := bits.RotateLeft32(e, -6) ^ bits.RotateLeft32(e, -11) ^ bits.RotateLeft32(e, -25)

ch := (e & f) ^ ((^e) & g)

tmp1 := h + S1 + ch + K[i] + w[i]

S0 := bits.RotateLeft32(a, -2) ^ bits.RotateLeft32(a, -13) ^ bits.RotateLeft32(a, -22)

maj := (a & b) ^ (a & c) ^ (b & c)

tmp2 := S0 + maj

h = g

g = f

f = e

e = d + tmp1

d = c

c = b

b = a

a = tmp1 + tmp2

}

h0 += a

h1 += b

h2 += c

h3 += d

h4 += e

h5 += f

h6 += g

h7 += h

}

digest := [32]byte{}

binary.BigEndian.PutUint32(digest[:4], h0)

binary.BigEndian.PutUint32(digest[4:8], h1)

binary.BigEndian.PutUint32(digest[8:12], h2)

binary.BigEndian.PutUint32(digest[12:16], h3)

binary.BigEndian.PutUint32(digest[16:20], h4)

binary.BigEndian.PutUint32(digest[20:24], h5)

binary.BigEndian.PutUint32(digest[24:28], h6)

binary.BigEndian.PutUint32(digest[28:], h7)

return digest

}

**Фрагмент основної програми:**

func main() {

msg := []byte("hello world")

hashed1 := sha256.Hash(msg)

hashed2 := sha256pkg.Sum256(msg)

if hashed2 != hashed1 {

fmt.Println("Hash codes is not matched")

} else {

fmt.Println("Hash codes is matched")

}

fmt.Println(hashed1)

fmt.Println(hashed2)

}

**Результати роботи програми**

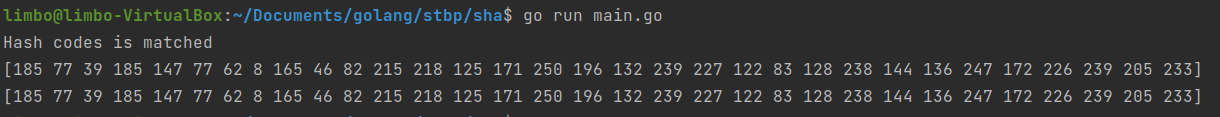
****

Рисунок 1 – Результат роботи програми

**Висновки:** досліджено і реалізувано механізм хешування алгоритму SHA256. Розроблено програму роботи даного алгоритму.