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»

**«Симетричне шифрування. Алгоритм AES»**

Виконав: студент групи КН-М922б

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

(прізвище, ініціали)

Перевірив: доцент кафедри

\_Бульба С. С \_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Харків - 2022

**Мета:** Дослідити принципи роботи симетричного шифрування на прикладі алгоритму AES.

**Завдання:**

Реалізувати алгоритм симетричного шифрування AES (будь-якої версії - 128 або 256).

Довести коректність роботи реалізованого алгоритму шляхом порівняння результатів з існуючими реалізаціями (напр. сайтом-утилітою https://cryptii.com).

**Хід роботи**

**Ключові елементи програми:**

type AES struct {

nr int

nk int

nb int

len int

key []byte

roundKeys []uint32

}

func NewAES(key []byte) (\*AES, error) {

var nk, nr int

switch len(key) {

case 16:

nk = 4

nr = 10

case 24:

nk = 6

nr = 12

case 32:

nk = 8

nr = 14

default:

return nil, errors.New("invalid key length")

}

aes := AES{

nr: nr,

nk: nk,

nb: 4,

len: 16,

key: key,

}

aes.roundKeys = aes.keyExpansion()

return &aes, nil

}

func (a \*AES) EncryptCTR(in []byte, iv []byte) []byte {

ivTmp := make([]byte, len(iv))

copy(ivTmp, iv)

plainTmp := make([]byte, len(in))

copy(plainTmp, in)

ivNumber := big.NewInt(0).SetBytes(iv)

one := big.NewInt(1)

i := 0

for ; i < len(plainTmp)-a.len; i += a.len {

a.encryptBlock(ivTmp, a.roundKeys)

Xor(plainTmp[i:i+a.len], ivTmp)

ivNumber.Add(ivNumber, one).FillBytes(ivTmp)

}

a.encryptBlock(ivTmp, a.roundKeys)

Xor(plainTmp[i:], ivTmp)

return plainTmp

}

func (a \*AES) DecryptCTR(in []byte, iv []byte) []byte {

ivTmp := make([]byte, len(iv))

copy(ivTmp, iv)

cipherTmp := make([]byte, len(in))

copy(cipherTmp, in)

ivNumber := big.NewInt(0).SetBytes(iv)

one := big.NewInt(1)

i := 0

for ; i < len(cipherTmp)-a.len; i += a.len {

a.encryptBlock(ivTmp, a.roundKeys)

Xor(cipherTmp[i:i+a.len], ivTmp)

ivNumber.Add(ivNumber, one).FillBytes(ivTmp)

}

a.encryptBlock(ivTmp, a.roundKeys)

Xor(cipherTmp[i:], ivTmp)

return cipherTmp

}

**Фрагмент основної програми:**

func main() {

key := []byte{0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6, 0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c}

\_aes, err := aes.NewAES(key)

if err != nil {

log.Fatal(err)

}

iv := []byte{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f}

enc := \_aes.EncryptCTR([]byte("hello"), iv)

for \_, c := range enc {

fmt.Printf("%x ", c)

}

fmt.Println()

dec := \_aes.DecryptCTR(enc, iv)

fmt.Println(string(dec))

}

**Результати роботи програми**

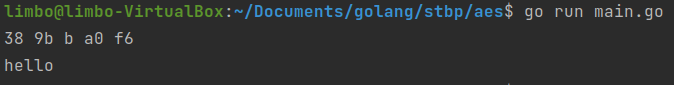
****

Рисунок 1 – Результат роботи програми

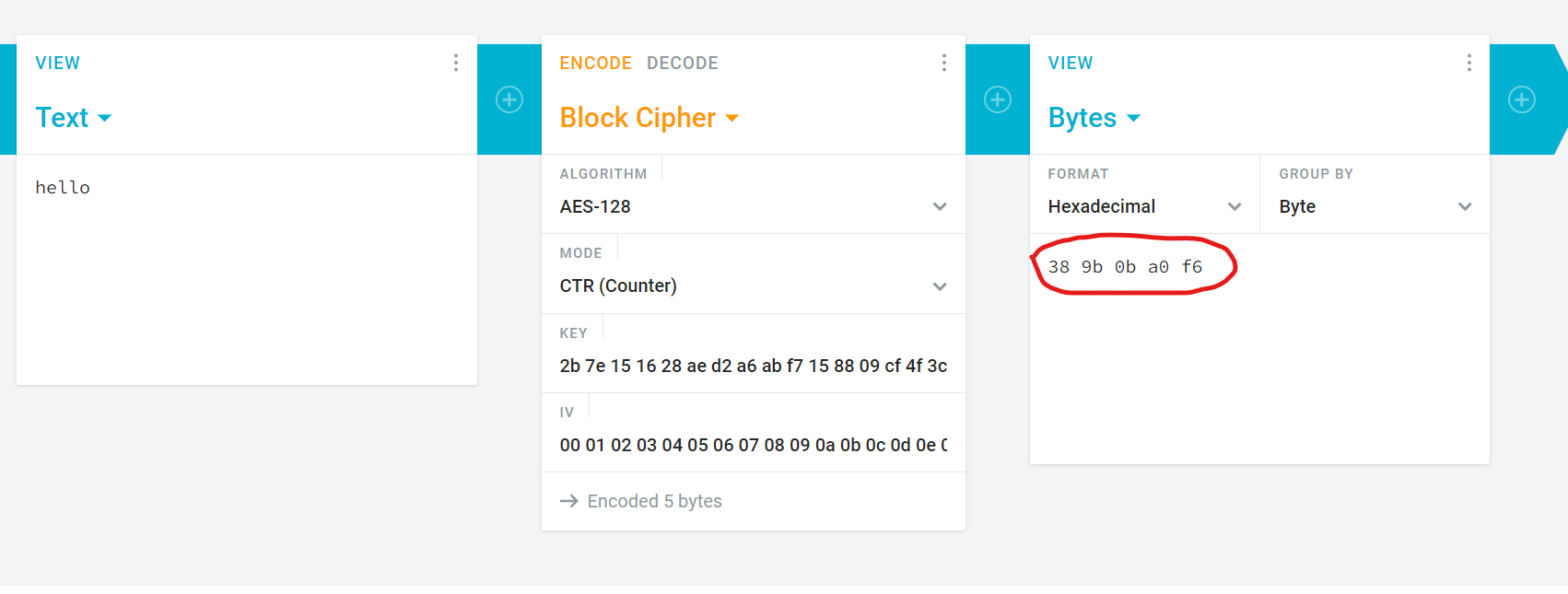


Рисунок 2 – Перевірка результату за допомогою веб додатку cryptii.com

**Висновки:** досліджено принципи роботи симетричного шифрування на прикладі алгоритму AES. Створено програму, яка реалізує алгоритм AES.