Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №7»**

“Исследование блочных шифров”

**Выполнил:** студент 3 курса

4 группы специальности ПОИТ

Зворыкин Дмитрий Александрович

Минск 2023

**Вариант 9 (4) – DES EEE2**

Исходный код:

|  |
| --- |
| package main  import (  "bytes"  "crypto/cipher"  "crypto/des"  "encoding/hex"  "fmt"  "time"  )  func main() {  plaintext := []byte("Hello, world! 1234567890")  key := []byte("securitysecuritysecurity")  block, err := des.NewTripleDESCipher(key)  if err != nil {  panic(err)  }  blockSize := block.BlockSize()  plaintext = addPadding(plaintext, blockSize)  ciphertext := make([]byte, len(plaintext))  mode := cipher.NewCBCEncrypter(block, key[:blockSize])  startEncrypt := time.Now()  mode.CryptBlocks(ciphertext, plaintext)  elapsedEncrypt := time.Since(startEncrypt)  decryptedText := make([]byte, len(ciphertext))  mode = cipher.NewCBCDecrypter(block, key[:blockSize])  startDecrypt := time.Now()  mode.CryptBlocks(decryptedText, ciphertext)  elapsedDecrypt := time.Since(startDecrypt)  decryptedText = removePadding(decryptedText)  var diffBits int  for i := 0; i < len(plaintext); i++ {  xorByte := plaintext[i] ^ ciphertext[i]  for j := 0; j < 8; j++ {  if (xorByte & (1 << j)) != 0 {  diffBits++  }  }  }  fmt.Printf("Plaintext: %s\n", plaintext)  fmt.Printf("Ciphertext: %x\n", ciphertext)  fmt.Printf("Decrypted text: %s\n", decryptedText)  fmt.Printf("Encryption time: %s\n", elapsedEncrypt)  fmt.Printf("Decryption time: %s\n", elapsedDecrypt)  fmt.Printf("Avalanche effect: %d bits changed\n", diffBits)  analyzeTripleDesWeakKeys()  }  func addPadding(data []byte, blockSize int) []byte {  padding := blockSize - (len(data) % blockSize)  padtext := bytes.Repeat([]byte{byte(padding)}, padding)  return append(data, padtext...)  }  func removePadding(data []byte) []byte {  padding := int(data[len(data)-1])  return data[:len(data)-padding]  }  func analyzeTripleDesWeakKeys() {  weakKeys := []string{  "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",  "000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF",  "FFFFFFFFFFFFFFFFFFFFFFFF000000000000000000000000",  }  semiWeakKeys := []string{  "011F011F011F011F22E522E522E522E522E522E522E522E5",  "1F011F010E011E010F22E522FE22E5220F22E522FE22E522",  "E001E001F101F101C122C122C122C122C122C122C122C122",  }  fmt.Println("Analysis of Triple DES weak and semi-weak keys:")  fmt.Println("==================================================")  for i := range weakKeys {  key := weakKeys[i]  fmt.Printf("Weak key %d: %s\n", i+1, key)  result := analyzeTripleDesKey(key)  fmt.Printf("Changed bits: %d\n", result.changedBits)  fmt.Printf("Average changed bits: %.2f\n", result.changedBitsAvg)  }  for i := range semiWeakKeys {  key := semiWeakKeys[i]  fmt.Printf("Semi-weak key %d: %s\n", i+1, key)  result := analyzeTripleDesKey(key)  fmt.Printf("Changed bits: %d\n", result.changedBits)  fmt.Printf("Average changed bits: %.2f\n", result.changedBitsAvg)  }  }  type tripleDesKeyAnalysisResult struct {  changedBits int  changedBitsAvg float64  }  func analyzeTripleDesKey(key string) tripleDesKeyAnalysisResult {  keyBytes, err := hex.DecodeString(key)  if err != nil {  panic(err)  }  block, err := des.NewTripleDESCipher(keyBytes)  if err != nil {  panic(err)  }  blockSize := block.BlockSize()  plaintext := make([]byte, blockSize)  ciphertext := make([]byte, blockSize)  mode := cipher.NewCBCEncrypter(block, keyBytes[:blockSize])  mode.CryptBlocks(ciphertext, plaintext)  var diffBits int  for i := 0; i < blockSize; i++ {  xorByte := plaintext[i] ^ ciphertext[i]  for j := 0; j < 8; j++ {  if (xorByte & (1 << j)) != 0 {  diffBits++  }  }  }  return tripleDesKeyAnalysisResult{  changedBits: diffBits,  changedBitsAvg: float64(diffBits) / float64(blockSize),  }  } |

Результат выполнения:

