



**Министерство науки и высшего образования Российской  
Федерации**  
**Федеральное государственное бюджетное образовательное  
учреждение высшего образования**  
**«Московский государственный технический университет  
имени Н.Э. Баумана**  
**(национальный исследовательский университет)»**  
**(МГТУ им. Н.Э. Баумана)**

---

**ФАКУЛЬТЕТ «Информатика и системы управления»**

**КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»**

**Лабораторная работа № 7**  
**по дисциплине «Защита информации»**

**Тема Реализация алгоритма симметричного шифрования (AES)**

**Студент Пермякова Е. Д.**

**Группа ИУ7-72Б**

**Преподаватели Руденкова Ю. С.**

Москва, 2025

## **ВВЕДЕНИЕ**

Целью данной работы является разработка алгоритма симметричного шифрования (AES). Шифрование и расшифровка произвольного файла.

Для достижения поставленной цели требуется решить следующие задачи:

- 1) описать алгоритм симметричного шифрования (AES);
- 2) реализовать в виде программы алгоритм симметричного шифрования (AES);

# 1 Теоретическая часть

Виды симметричного шифрования:

- 1) Поточные: Шифруют данные побитово/побайтово (RC4);
- 2) Блочные: Шифруют данные блоками фиксированного размера (DES, AES);

Алгоритмы перестановки — это методы, которые изменяют порядок следования элементов (но не их значения) посредством присваивания и перестановки их значений. Пример: IP в DES.

Алгоритмы подстановки — это методы шифрования, в которых элементы исходного открытого текста заменяются зашифрованным текстом в соответствии с некоторым правилом. Пример: шифр Цезаря.

Алгоритм DES использует методы перестановки и подстановки.

## 2 Описание алгоритма симметричного шифрования (AES)

На рисунке 2.1 приведена схема алгоритма симметричного шифрования (AES).



Рисунок 2.1 – Схема алгоритма симметричного шифрования (AES)

### 3 Пример работы алгоритма симметричного шифрования (AES)

На рисунке 3.1 приведен пример работы алгоритма симметричного шифрования (AES).

```
⊗ kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go
Использование:
  Шифрование: go run aes.go encrypt <файл_ключа> <входной_файл> <выходной_файл>
  Дешифрование: go run aes.go decrypt <файл_ключа> <входной_файл> <выходной_файл>
  Генерация ключа: go run aes.go genkey <файл_ключа> [128|192|256]
exit status 1
● kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go genkey ./k 128
Генерация ключа 128 бит в файл: ./k
Ключ успешно сгенерирован и сохранен в: ./k
Размер ключа: 16 байт (128 бит)
SHA256 (первые 16 байт): 026cdd942ffeedd1ac3927be49f08145
● kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go encrypt ./k ../data/text.txt
../data/encrypted.txt
Шифрование файла: ../data/text.txt -> ../data/encrypted.txt
Используется ключ: ./k (128 бит)
Шифрование завершено успешно!
Размер исходного файла: 1904 байт
Размер зашифрованного файла: 1920 байт
● kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go decrypt ./k ../data/encrypted
.txt ../data/decrypted.txt
Дешифрование файла: ../data/encrypted.txt -> ../data/decrypted.txt
Используется ключ: ./k (128 бит)
Дешифрование завершено успешно!
Размер зашифрованного файла: 1920 байт
Размер расшифрованного файла: 1894 байт
```

Рисунок 3.1 – Пример работы алгоритма симметричного шифрования (AES)

На рисунках 3.2-3.4 приведено содержимое исходного, зашифрованного и расшифрованного файлов.

```
kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ cat ../data/text.txt
– Ну, здравствуйте, здравствуйте. Je vois que je vous fais peur 2, садитесь и рассказывайте.
Так говорила в июле 1805 года известная Анна Павловна Шерер, фрейлина и приближенная императрицы Марии Феодоровны, встречая важного и чиновного князя Василия, первого приехавшего на ее вечер. Анна Павловна кашляла несколько дней, у нее был грипп, как она говорила (грипп был тогда новое слово, употреблявшееся только редкими). В записочках, разосланных утром с красным лакеем, было написано без различия во всех:
«Si vous n'avez rien de mieux à faire, Monsieur le comte (или mon prince), et si la perspective de passer la soirée chez une pauvre malade ne vous effraye pas trop, je serai charmée de vous voir chez moi entre 7 et 10 heures. Annette Scherer» 3.
– Dieu, quelle virulente sortie! 4 – отвечал, несколько не смутясь такою встречей, вошедший князь, в придворном, шитом мундире, в чулках, башмаках и звездах, с светлым выражением плоского лица.
Он говорил на том изысканном французском языке, на котором не только говорили, но и думали наши деда, и с теми, тихими, покровительственными интонациями, которые свойственны состаревшемуся в свете и при дворе значительному человеку.
kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ cat ../data/encrypted.txt
[FqP0%EpL2k]
          tI000BlJ000Fl<)YVj00400[(000x0Y&0_S/00{0R0#-'00^m00000~
                                                                KN00t
00I006c%/0\000W00B00h2|00V0MoZE00000z0.x>0000x0\000Y4{0
                                                                aT
0"D8*g00xG00005auE080(0 %000[7010W0Y]Y0000g00
0_0000U0Y"0j00TjL-0T0n0i0N0
=00V0e00Nc00000E00!t0@000#0000:0
```

Рисунок 3.2 – Содержимое исходного и зашифрованного файлов

```
kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ cat ../data/decrypted.txt
– Ну, здравствуйте, здравствуйте. Je vois que je vous fais peur 2, садитесь и рассказывайте.
Так говорила в июле 1805 года известная Анна Павловна Шерер, фрейлина и приближенная императрицы Марии Феодоровны, встречая важного и чиновного князя Василия, первого приехавшего на ее вечер. Анна Павловна кашляла несколько дней, у нее был грипп, как она говорила (грипп был тогда новое слово, употреблявшееся только редкими). В записочках, разосланных утром с красным лакеем, было написано без различия во всех:
«Si vous n'avez rien de mieux à faire, Monsieur le comte (или mon prince), et si la perspective de passer la soirée chez une pauvre malade ne vous effraye pas trop, je serai charmée de vous voir chez moi entre 7 et 10 heures. Annette Scherer» 3.
– Dieu, quelle virulente sortie! 4 – отвечал, несколько не смутясь такою встречей, вошедший князь, в придворном, шитом мундире, в чулках, башмаках и звездах, с светлым выражением плоского лица.
Он говорил на том изысканном французском языке, на котором не только говорили, но и думали наши деда, и с теми, тихими, покровительственными интонациями, которые свойственны состаревшемуся в свете и при дворе значительному человеку.
```

Рисунок 3.3 – Содержимое расшифрованного файла

На рисунке 3.4 приведен пример шифрования и расшифровки при попытке расшифровки файла сторонним ключом.

```

kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go genkey ./k
Генерация ключа 256 бит в файл: ./k
Внимание: Файл ./k уже существует. Перезаписать? (y/N): y
Ключ успешно сгенерирован и сохранен в: ./k
Размер ключа: 32 байт (256 бит)
SHA256 (первые 16 байт): 9995ebe25f25dee1795d6cae93aad05809cc6cd0e9f38db933f7b99e165aa3c7
kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ go run aes.go decrypt ./k ../data/encrypted.txt ../data/decrypted.txt
Дешифрование файла: ../data/encrypted.txt -> ../data/decrypted.txt
Используется ключ: ./k (256 бит)
Дешифрование завершено успешно!
Размер зашифрованного файла: 1920 байт
Размер расшифрованного файла: 1904 байт
kathrine@Viva:~/vuz/InfoSec/is_7_aes/src$ cat ../data/decrypted.txt
H#1«»>*h0x$00.41)gzTRKSS4tnj
P h 2 + Y
JeH, :5

```

Рисунок 3.4 – Пример шифрования и расшифровки при попытке расшифровки файла сторонним ключом

## 4 Реализация алгоритма симметричного шифрования (AES)

В качестве средства реализации алгоритма симметричного шифрования (AES) был выбран язык Go.

```
package main

import (
    "crypto/rand"
    "errors"
    "fmt"
    "io"
    "os"
    "strconv"
)

const (
    AESBlockSize      = 16
    AESStateDim       = 4
    AESRounds128      = 10
    AESRounds192      = 12
    AESRounds256      = 14
    BitsPerByte       = 8
    WordSize          = 4
    GFReducingPoly    = 0x1B
    GFMSBMask         = 0x80
)

type AESKeySize int

const (
    AESKeySize128 AESKeySize = 16
    AESKeySize192 AESKeySize = 24
    AESKeySize256 AESKeySize = 32
)

type AESError int

const (
    AESSuccess AESError = iota
    AESErrorUnsupportedKeySize
    AESErrorMemoryAllocation
)
```



```

AESErrorInvalidInput
)

func (e AESError) String() string {
    switch e {
        case AESSuccess:
            return "Success"
        case AESErrorUnsupportedKeySize:
            return "Unsupported_key_size"
        case AESErrorMemoryAllocation:
            return "Memory_allocation_failed"
        case AESErrorInvalidInput:
            return "Invalid_input_parameters"
        default:
            return "Unknown_error"
    }
}

var sbox = [256]byte{
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01,
        0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4,
        0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5,
        0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12,
        0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b,
        0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb,
        0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9,
        0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6,
        0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7,
        0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee,
        0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3,
        0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,

```

```

    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56,
        0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd,
        0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35,
        0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e,
        0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99,
        0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16,
}

```

```

var rsbox = [256]byte{
    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40,
        0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
    0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e,
        0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
    0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c,
        0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b,
        0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4,
        0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15,
        0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4,
        0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
    0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf,
        0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
    0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2,
        0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9,
        0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
    0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7,
        0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb,
        0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12,
        0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5,
        0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
}

```

```

    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb,
        0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69,
        0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d,
}

var rcon = [32]byte{
    0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
        0x36,
    0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6,
        0x97,
    0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
}

type AESState [AESStateDim][AESStateDim]byte

func secureZeroMemory(data []byte) {
    for i := range data {
        data[i] = 0
    }
}

func wordRotateLeft(word []byte) {
    if len(word) != WordSize {
        return
    }
    temp := word[0]
    word[0] = word[1]
    word[1] = word[2]
    word[2] = word[3]
    word[3] = temp
}

func keyScheduleCore(word []byte, iteration byte) {
    wordRotateLeft(word)
    for i := 0; i < WordSize; i++ {
        word[i] = sbbox[word[i]]
    }
    word[0] ^= rcon[iteration]
}

```

```

func aesExpandKey(key []byte, keySize AESKeySize) ([]byte,
error) {
    var numRounds int
    switch keySize {
        case AESKeySize128:
            numRounds = AESRounds128
        case AESKeySize192:
            numRounds = AESRounds192
        case AESKeySize256:
            numRounds = AESRounds256
        default:
            return nil, errors.New("unsupported_key_size")
    }

    expandedKeySize := AESBlockSize * (numRounds + 1)
    expandedKey := make([]byte, expandedKeySize)
    copy(expandedKey, key)

    currentSize := len(key)
    rconIteration := byte(1)
    tempWord := make([]byte, WordSize)

    for currentSize < expandedKeySize {
        copy(tempWord, expandedKey[currentSize-WordSize:
            currentSize])

        if currentSize%int(keySize) == 0 {
            keyScheduleCore(tempWord, rconIteration)
            rconIteration++
        }

        if keySize == AESKeySize256 && currentSize%int(keySize)
            == AESBlockSize {
            for i := 0; i < WordSize; i++ {
                tempWord[i] = sbox[tempWord[i]]
            }
        }

        for i := 0; i < WordSize; i++ {
            expandedKey[currentSize] = expandedKey[
                currentSize-int(keySize)] ^ tempWord[i]

```

```

        currentSize++
    }
}

return expandedKey, nil
}

func subBytes(state *AESState) {
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            state[r][c] = sbox[state[r][c]]
        }
    }
}

func invSubBytes(state *AESState) {
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            state[r][c] = rsbox[state[r][c]]
        }
    }
}

func shiftRows(state *AESState) {
    // Row 1: 1-byte left shift
    temp := state[1][0]
    state[1][0] = state[1][1]
    state[1][1] = state[1][2]
    state[1][2] = state[1][3]
    state[1][3] = temp

    // Row 2: 2-byte left shift
    temp = state[2][0]
    state[2][0] = state[2][2]
    state[2][2] = temp
    temp = state[2][1]
    state[2][1] = state[2][3]
    state[2][3] = temp

    // Row 3: 3-byte left shift
    temp = state[3][0]

```

```

    state[3][0] = state[3][3]
    state[3][3] = state[3][2]
    state[3][2] = state[3][1]
    state[3][1] = temp
}

func invShiftRows(state *AESState) {
    // Row 1: 1-byte right shift
    temp := state[1][3]
    state[1][3] = state[1][2]
    state[1][2] = state[1][1]
    state[1][1] = state[1][0]
    state[1][0] = temp

    // Row 2: 2-byte right shift
    temp = state[2][0]
    state[2][0] = state[2][2]
    state[2][2] = temp
    temp = state[2][1]
    state[2][1] = state[2][3]
    state[2][3] = temp

    // Row 3: 3-byte right shift
    temp = state[3][3]
    state[3][3] = state[3][0]
    state[3][0] = state[3][1]
    state[3][1] = state[3][2]
    state[3][2] = temp
}

func galoisMul(a, b byte) byte {
    p := byte(0)
    for i := 0; i < BitsPerByte; i++ {
        if b&1 != 0 {
            p ^= a
        }

        hiBitSet := a & GFMSBMask
        a <<= 1

        if hiBitSet != 0 {

```

```

        a ^= GFReducingPoly
    }

    b >>= 1
}
return p
}

func mixColumns(state *AESState) {
    var t [AESStateDim]byte
    for c := 0; c < AESStateDim; c++ {
        for r := 0; r < AESStateDim; r++ {
            t[r] = state[r][c]
        }

        state[0][c] = galoisMul(t[0], 2) ^ galoisMul(t[1], 3) ^
            t[2] ^ t[3]
        state[1][c] = t[0] ^ galoisMul(t[1], 2) ^ galoisMul(t
            [2], 3) ^ t[3]
        state[2][c] = t[0] ^ t[1] ^ galoisMul(t[2], 2) ^
            galoisMul(t[3], 3)
        state[3][c] = galoisMul(t[0], 3) ^ t[1] ^ t[2] ^
            galoisMul(t[3], 2)
    }
}

func invMixColumns(state *AESState) {
    var t [AESStateDim]byte
    for c := 0; c < AESStateDim; c++ {
        for r := 0; r < AESStateDim; r++ {
            t[r] = state[r][c]
        }

        state[0][c] = galoisMul(t[0], 14) ^ galoisMul(t[1], 11)
            ^ galoisMul(t[2], 13) ^ galoisMul(t[3], 9)
        state[1][c] = galoisMul(t[0], 9) ^ galoisMul(t[1], 14)
            ^ galoisMul(t[2], 11) ^ galoisMul(t[3], 13)
        state[2][c] = galoisMul(t[0], 13) ^ galoisMul(t[1], 9)
            ^ galoisMul(t[2], 14) ^ galoisMul(t[3], 11)
        state[3][c] = galoisMul(t[0], 11) ^ galoisMul(t[1], 13)
            ^ galoisMul(t[2], 9) ^ galoisMul(t[3], 14)
    }
}

```

```

    }
}

func addRoundKey(state *AESState, roundKey []byte) {
    for c := 0; c < AESStateDim; c++ {
        for r := 0; r < AESStateDim; r++ {
            state[r][c] ^= roundKey[c*AESStateDim+r]
        }
    }
}

func aesEncryptBlock(plaintext []byte, key []byte, keySize
AESKeySize) ([]byte, error) {
    if len(plaintext) != AESBlockSize {
        return nil, errors.New("plaintext_must_be_16_bytes")
    }

    var numRounds int
    switch keySize {
        case AESKeySize128:
            numRounds = AESRounds128
        case AESKeySize192:
            numRounds = AESRounds192
        case AESKeySize256:
            numRounds = AESRounds256
        default:
            return nil, errors.New("unsupported_key_size")
    }

    expandedKey, err := aesExpandKey(key, keySize)
    if err != nil {
        return nil, err
    }
    defer secureZeroMemory(expandedKey)

    var state AESState
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            state[r][c] = plaintext[r+AESStateDim*c]
        }
    }
}

```



```

    addRoundKey(&state, expandedKey)
    for round := 1; round < numRounds; round++ {
        subBytes(&state)
        shiftRows(&state)
        mixColumns(&state)
        addRoundKey(&state, expandedKey[AESBlockSize*round:])
    }
    subBytes(&state)
    shiftRows(&state)
    addRoundKey(&state, expandedKey[AESBlockSize*numRounds:])

    ciphertext := make([]byte, AESBlockSize)
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            ciphertext[r+AESStateDim*c] = state[r][c]
        }
    }

    return ciphertext, nil
}

func aesDecryptBlock(ciphertext []byte, key []byte, keySize
AESKeySize) ([]byte, error) {
    if len(ciphertext) != AESBlockSize {
        return nil, errors.New("ciphertext must be 16 bytes")
    }

    var numRounds int
    switch keySize {
        case AESKeySize128:
            numRounds = AESRounds128
        case AESKeySize192:
            numRounds = AESRounds192
        case AESKeySize256:
            numRounds = AESRounds256
        default:
            return nil, errors.New("unsupported key size")
    }

    expandedKey, err := aesExpandKey(key, keySize)

```

```

    if err != nil {
        return nil, err
    }
    defer secureZeroMemory(expandedKey)

    var state AESState
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            state[r][c] = ciphertext[r+AESStateDim*c]
        }
    }

    addRoundKey(&state, expandedKey[AESBlockSize*numRounds:])
    for round := numRounds; round > 1; round-- {
        invShiftRows(&state)
        invSubBytes(&state)
        addRoundKey(&state, expandedKey[AESBlockSize*(round-1):])
        invMixColumns(&state)
    }
    invShiftRows(&state)
    invSubBytes(&state)
    addRoundKey(&state, expandedKey)

    plaintext := make([]byte, AESBlockSize)
    for r := 0; r < AESStateDim; r++ {
        for c := 0; c < AESStateDim; c++ {
            plaintext[r+AESStateDim*c] = state[r][c]
        }
    }

    return plaintext, nil
}

func encryptFile(inputPath, outputPath string, key []byte,
    keySize AESKeySize) error {
    inputFile, err := os.Open(inputPath)
    if err != nil {
        return err
    }
    defer inputFile.Close()

```

```

outputFile, err := os.Create(outputPath)
if err != nil {
    return err
}
defer outputFile.Close()

// IV (Initialization Vector)
iv := make([]byte, AESBlockSize)
if _, err := rand.Read(iv); err != nil {
    return err
}

if _, err := outputFile.Write(iv); err != nil {
    return err
}

buffer := make([]byte, AESBlockSize)
previousBlock := make([]byte, AESBlockSize)
copy(previousBlock, iv)

for {
    n, err := inputFile.Read(buffer)
    if err != nil && err != io.EOF {
        return err
    }

    if n == 0 {
        break
    }

    if n < AESBlockSize {
        padding := byte(AESBlockSize - n)
        for i := n; i < AESBlockSize; i++ {
            buffer[i] = padding
        }
    }

    for i := 0; i < AESBlockSize; i++ {
        buffer[i] ^= previousBlock[i]
    }
}

```

```

        encryptedBlock, err := aesEncryptBlock(buffer, key,
            keySize)
        if err != nil {
            return err
        }

        if _, err := outputFile.Write(encryptedBlock); err !=
            nil {
            return err
        }

        copy(previousBlock, encryptedBlock)
    }

    return nil
}

func decryptFile(inputPath, outputPath string, key []byte,
    keySize AESKeySize) error {
    inputFile, err := os.Open(inputPath)
    if err != nil {
        return err
    }
    defer inputFile.Close()

    outputFile, err := os.Create(outputPath)
    if err != nil {
        return err
    }
    defer outputFile.Close()

    iv := make([]byte, AESBlockSize)
    if _, err := inputFile.Read(iv); err != nil {
        return err
    }

    fileInfo, err := inputFile.Stat()
    if err != nil {
        return err
    }

```

```

fileSize := fileInfo.Size()
if fileSize%AESBlockSize != 0 || fileSize < int64(
    AESBlockSize) {
    return errors.New("invalid_encrypted_file_size")
}

previousBlock := make([]byte, AESBlockSize)
copy(previousBlock, iv)
buffer := make([]byte, AESBlockSize)

for {
    n, err := inputFile.Read(buffer)
    if err != nil && err != io.EOF {
        return err
    }

    if n == 0 {
        break
    }

    decryptedBlock, err := aesDecryptBlock(buffer, key,
        keySize)
    if err != nil {
        return err
    }

    for i := 0; i < AESBlockSize; i++ {
        decryptedBlock[i] ^= previousBlock[i]
    }

    copy(previousBlock, buffer)

    bytesToWrite := AESBlockSize
    if inputFile, err := inputFile.Seek(0, io.SeekCurrent);
        err == nil {
        if inputFile == fileSize {
            padding := decryptedBlock[AESBlockSize-1]
            if padding > 0 && padding <= AESBlockSize {
                bytesToWrite = AESBlockSize - int(padding)
            }
        }
    }
}

```

```

        }
    }

    if _, err := outputFile.Write(decryptedBlock[:
        bytesToWrite]); err != nil {
        return err
    }
}

return nil
}

func generateKey(keySize AESKeySize) ([]byte, error) {
    key := make([]byte, int(keySize))
    if _, err := rand.Read(key); err != nil {
        return nil, err
    }
    return key, nil
}

func saveKeyToFile(key []byte, filename string) error {
    return os.WriteFile(filename, key, 0600)
}

func loadKeyFromFile(filename string) ([]byte, error) {
    key, err := os.ReadFile(filename)
    if err != nil {
        return nil, err
    }

    switch len(key) {
    case 16, 24, 32:
        return key, nil
    default:
        return nil, errors.New("invalid_key_size. Must be 16, 24, or 32 bytes")
    }
}

```

Листинг 4.1 – Реализация алгоритма симметричного шифрования (AES)

## **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы был реализован алгоритм симметричного шифрования (AES).

В процессе выполнения данной работы были выполнены все задачи:

- 1) описать алгоритм симметричного шифрования (AES);
- 2) реализовать в виде программы алгоритм симметричного шифрования (AES);