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 № 2
по дисциплине «Защита информации»

Тема Реализация электронного аналога шифровальной машины «Энигма»

Студент Пермякова Е. Д.

Группа ИУ7-72Б

Преподаватели Руденкова Ю. С.

Москва, 2025

ВВЕДЕНИЕ

Целью данной работы является разработка электронного аналога шифровальной машины, шифрование и расшифровка архивных файлов.

Для достижения поставленной цели требуется решить следующие задачи:

1) описать алгоритм работы электронного аналога шифровальной машины «Энигма»;

2) реализовать виде программы электронный аналог шифровальной машины «Энигма» для шифрования и расшифровки архивных файлов;

1 Теоретическая часть

Информация – это сведения, рассматриваемые в контексте их содержания.

Защита информации – это процесс предотвращения несанкционированного доступа, использования, раскрытия, разрушения или изменения данных.

Актив – это любой компонент информационной системы (данные, оборудование, программное обеспечение, персонал, услуги, репутация), который имеет ценность для организации и поэтому требует защиты.

Информационная сфера – это совокупность информации, информационной инфраструктуры, субъектов, осуществляющих сбор, формирование, распространение и использование информации, а также системы регулирования возникающих при этом общественных отношений.

Угроза – это потенциальная возможность того, что определенное лицо, действие, событие или явление (источник угрозы) преднамеренно или случайно нарушит безопасность информации (ее конфиденциальность, целостность, доступность), нанеся ущерб владельцу или пользователю информации.

Безопасность – это состояние защищенности жизненно важных интересов личности, общества и государства от внутренних и внешних угроз.

Информационная безопасность – это комплекс мер и средств, направленных на защиту конфиденциальности, целостности и доступности информации.

Шифровальная машина «Энигма» – это портативная электромеханическая шифровальная машина, использовавшаяся в XX веке (в основном нацистской Германией во Второй мировой войне) для защиты служебной переписки. Ее основным принципом работы было многоалфавитное шифрование с изменяющимся алфавитом замены после каждой буквы, реализуемое с помощью системы вращающихся роторов.

2 Описание алгоритма шифрования и расшифровки архивного файла

На вход электронному аналогу шифровальной машины «Энигма» подавался архив для его шифрования и расшифровки.

На рисунке 2.1 приведена схема работы электронного аналога шифровальной машины «Энигма»

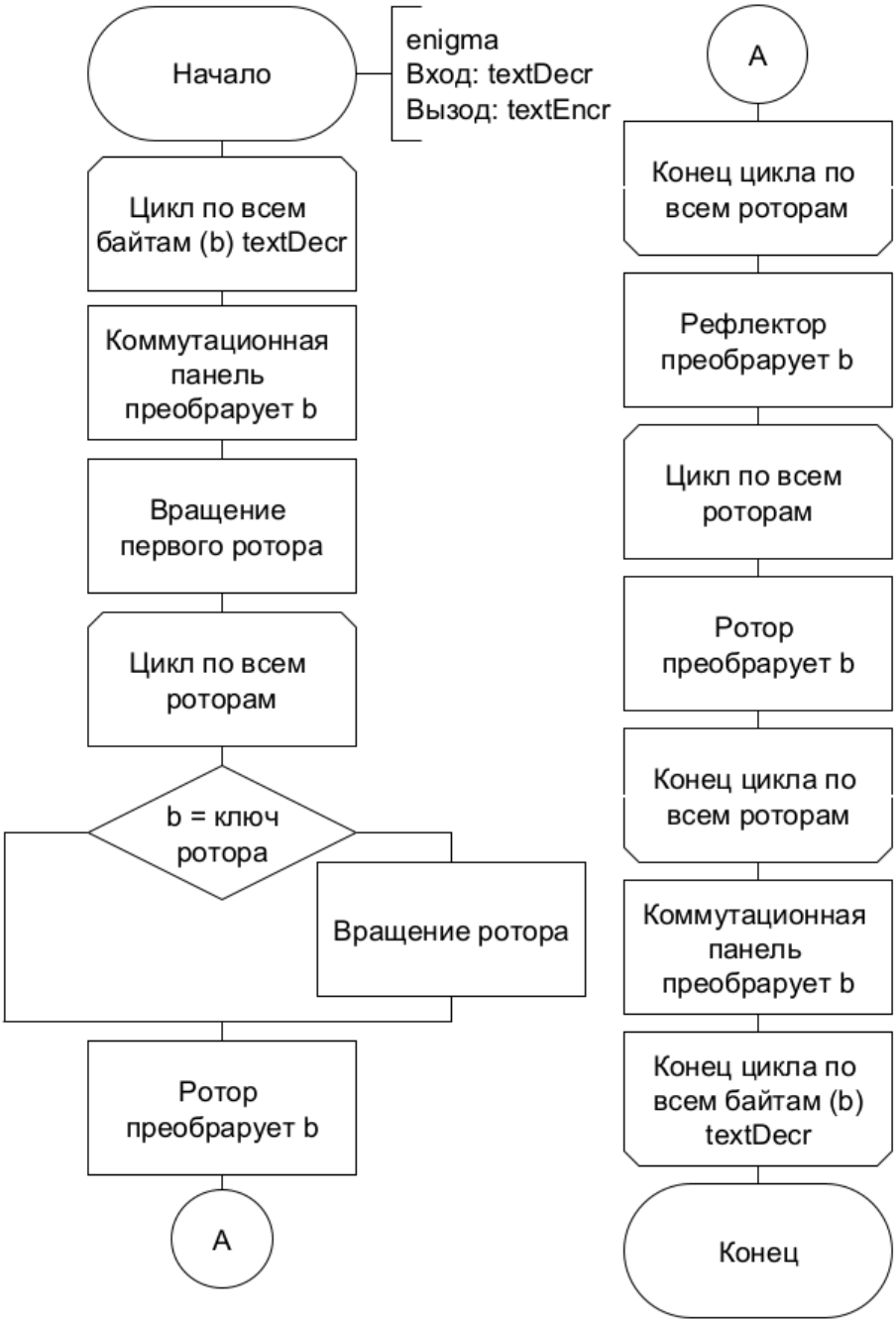

Рисунок 2.1 – Схема работы электронного аналога шифровальной машины «Энигма»

3 Пример работы электронного аналога шифровальной машины «Энигма»

Для архивирования и разархивирования файлов использовалась программа WinRAR [1].

На рисунках 3.1-3.4 приведен пример работы электронного аналога шифровальной машины «Энигма» для шифрования и расшифровки архивных файлов.

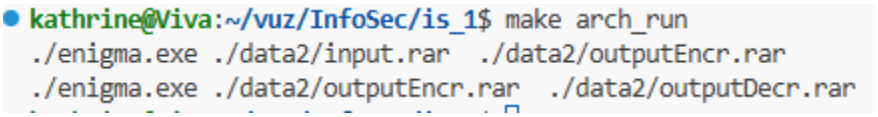

Рисунок 3.1 – Пример работы электронного аналога шифровальной машины «Энигма» – вызов программы

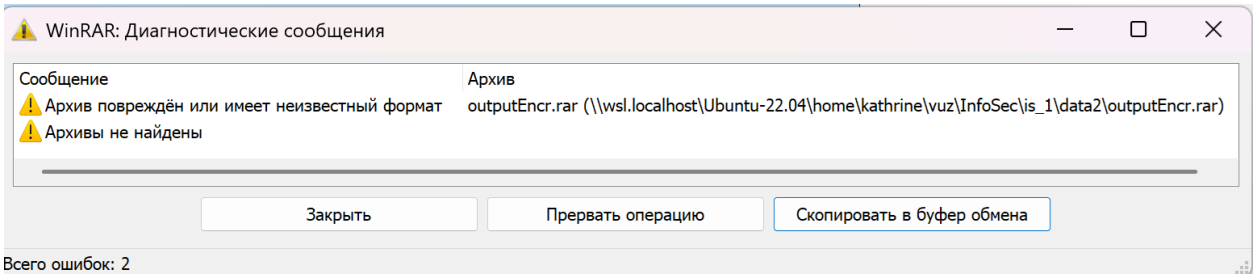

Рисунок 3.2 – Пример работы электронного аналога шифровальной машины «Энигма» – попытка разархивировать зашифрованный архив

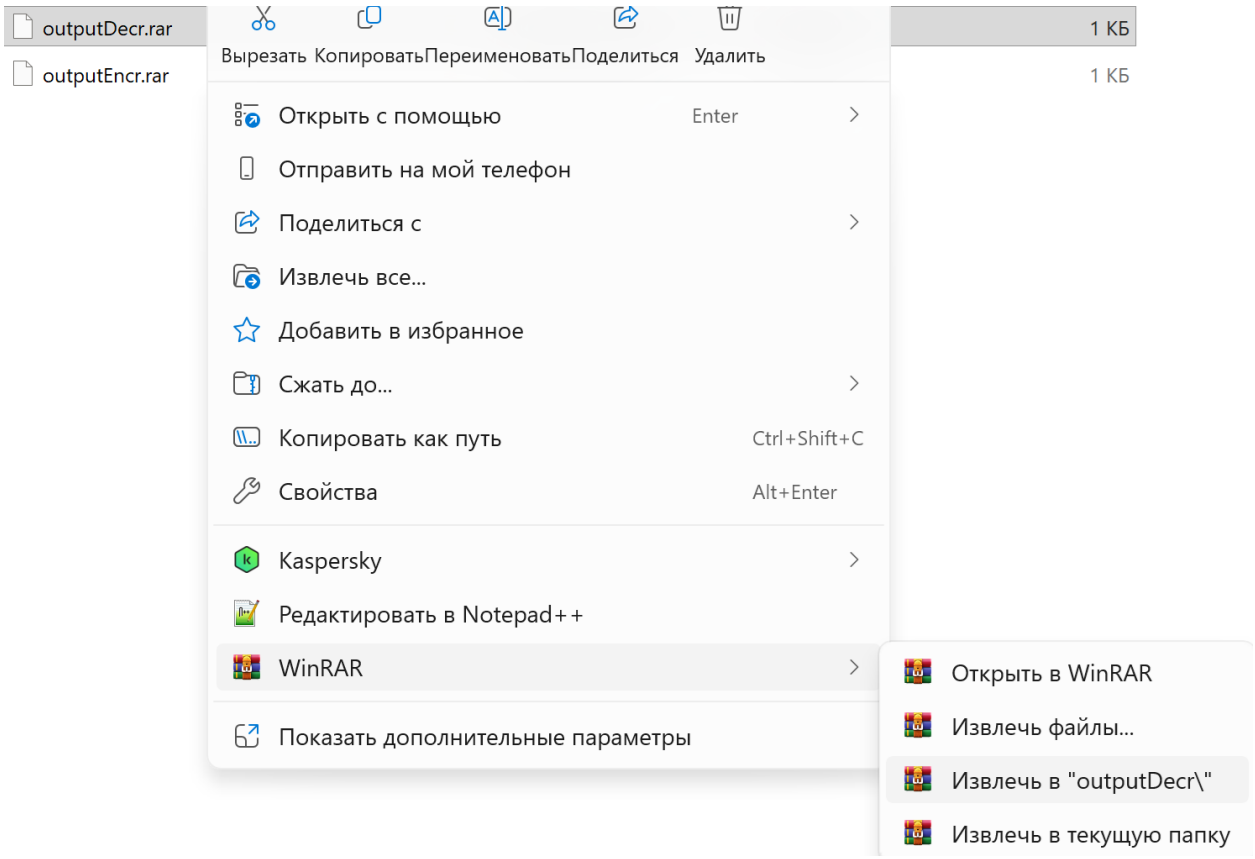

Рисунок 3.3 – Пример работы электронного аналога шифровальной машины «Энигма» – разархивирование расшифрованного архива

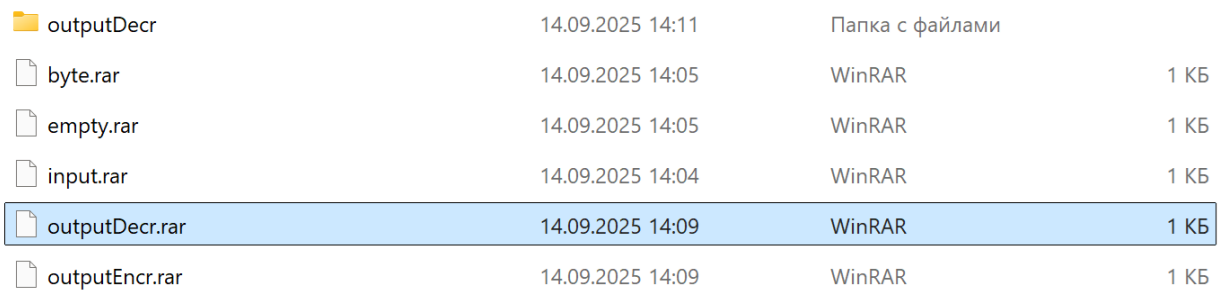

Рисунок 3.4 – Пример работы электронного аналога шифровальной машины «Энигма» – расшифрованный архив был успешно разархивирован

4 Реализация электронного аналога шифровальной машины «Энигма»

В качестве средства реализации электронного аналога шифровальной машины «Энигма» был выбран язык Go.

```
type Enigma interface {
    EncryptAlpha(alpha byte) byte
    EncryptText(text []byte) []byte
    SetRotorPositions(poses []byte) error
}

type enigma struct {
    switchingPanel Rotor
    rotors         []Rotor
    reflector      Reflector
}

func NewEnigma(switchingPanel Rotor, rotors []Rotor, reflector
   Reflector) Enigma {
    return &enigma{
        switchingPanel: switchingPanel,
        rotors:         rotors,
        reflector:      reflector,
    }
}

func (e *enigma) EncryptText(text []byte) []byte {
    resText := make([]byte, len(text))
    for i, v := range text {
        resText[i] = e.EncryptAlpha(v)
    }
    return resText
}

func (e *enigma) EncryptAlpha(alpha byte) byte {
    alpha = e.switchingPanel.SwitchTo(alpha)
    Nrotors := len(e.rotors)
    e.rotors[0].Rotate()
    nextA := e.rotors[0].Transform(alpha, 0)
    lastRing := e.rotors[0].GetRing()
```

```
    for i := 1; i < Nrotors; i++ {
        if e.rotors[i-1].GetRing() == e.rotors[i-1].
           GetSteppingPos() {
            e.rotors[i].Rotate()
        }
        nextA = e.rotors[i].Transform(nextA, lastRing)
        lastRing = e.rotors[i].GetRing()
    }

    nextA = e.reflector.Transform(nextA, lastRing, -1)

    lastRing = 0
    for i := len(e.rotors) - 1; i >= 0; i-- {
        nextA = e.rotors[i].TransformBack(nextA, lastRing)
        lastRing = e.rotors[i].GetRing()
    }
    nextA = byte((int(nextA) - int(lastRing) + alphabetSize) %
       alphabetSize)
    nextA = e.switchingPanel.SwitchFrom(nextA)

    return nextA
}

func (e *enigma) SetRotorPositions(poses []byte) error {
    if len(poses) != len(e.rotors) {
        return ErrLenPoses
    }
    for i := 0; i < len(e.rotors); i++ {
        e.rotors[i].SetRing(poses[i])
    }
    return nil
}

type Rotor interface {
    Rotate()
    Transform(alpha byte, nextRing byte) byte
    TransformBack(alpha byte, nextRing byte) byte
    SwitchTo(alpha byte) byte
    SwitchFrom(alpha byte) byte
    GetRing() byte
    SetRing(ring byte)
```

```
    GetSteppingPos() byte
}

type rotor struct {
    permutation   []byte
    rePermutation []byte
    ring          byte
    steppingPos   byte
}

func NewRotor(permutation []byte, steppingPos byte, ring byte)
   Rotor {
    rePermutation := make([]byte, len(permutation))
    for i, v := range permutation {
        rePermutation[int(v)] = byte(i)
    }
    return &rotor{
        permutation:   permutation,
        steppingPos:   steppingPos,
        rePermutation: rePermutation,
        ring:          ring,
    }
}

func (r *rotor) Rotate() {
    r.ring = byte((int(r.ring) + 1) % alphabetSize)
}

func (r *rotor) Transform(alpha byte, prevRing byte) byte {
    // fmt.Printf("Transform: alpha=%c, prevRing=%c\n", alpha+'
       A', prevRing+'A')
    a := int(alpha)
    pr := int(prevRing)
    intputAlpha := (a + (int(r.ring) - pr + alphabetSize)) %
       alphabetSize
    return r.permutation[intputAlpha]
}

func (r *rotor) TransformBack(alpha byte, nextRing byte) byte {
    // fmt.Printf("TransformBack: alpha=%c, nextRing=%c\n",
       alpha+'A', nextRing+'A')
```

```
    a := int(alpha)
    pr := int(nextRing)
    intputAlpha := (a - (pr - int(r.ring)) + alphabetSize) %
       alphabetSize
    return r.rePermutation[intputAlpha]
}

func (r *rotor) SwitchTo(alpha byte) byte {
    return r.permutation[alpha]
}

func (r *rotor) SwitchFrom(alpha byte) byte {
    return r.rePermutation[alpha]
}

type Reflector interface {
    Transform(alpha byte, nextRing byte, dir int) byte
}

type reflector struct {
    permutation []byte
}

func NewReflector(permutation []byte) Reflector {
    return &reflector{
        permutation: permutation,
    }
}

func (r *reflector) Transform(alpha byte, nextRing byte, dir
   int) byte {
    a := int(alpha)
    ring := int(nextRing)
    input := (a + dir*ring + alphabetSize) % alphabetSize
    return r.permutation[input]
}

const alphabetSize = 256

func main() {
    if len(os.Args) < 3 {
```

```
        fmt.Println("Usage: enigma input.txt output.txt")
        os.Exit(1)
    }
    inputFileName := os.Args[1]  // "../data/input.txt"
    outputFIlename := os.Args[2] // "../data/output.txt"

    inputData, err := os.ReadFile(inputFileName)
    if err != nil {
        fmt.Println(err)
        os.Exit(1)
    }

    switchingPanel := NewRotor(TypeRotor256_1, '0', 0)
    rotors := []Rotor{
        NewRotor(TypeRotor256_1, '1', 0),
        NewRotor(TypeRotor256_2, '-', 0),
        NewRotor(TypeRotor256_3, ' ', 0),
    }
    reflector := NewReflector(Reflector256_2)
    enigm := NewEnigma(switchingPanel, rotors, reflector)

    posRing := []byte{'Q', '8', '8'}
    enigm.SetRotorPositions(posRing)
    encryptedText := enigm.EncryptText(inputData)
    err = os.WriteFile(outputFIlename, encryptedText, 0666)
    if err != nil {
        fmt.Println(err)
        os.Exit(1)
    }
}
```

Листинг 4.1 – Реализация электронного аналога шифровальной машины «Энигма»

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был реализован электронный аналога шифровальной машина «Энигма».

В процессе выполнения данной работы были выполнены все задачи:

1) описать алгоритм работы электронного аналога шифровальной машины «Энигма»;

2) реализовать виде программы электронный аналог шифровальной машины «Энигма» для шифрования и расшифровки архивных файлов;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. WinRAR. URL: Режим доступа: https://www.win-rar.com/start.html?&L=4 (дата обращения: 14.09.2025).