Отчет по лабораторной работе
“Блочное симметричное шифрование”
по дисциплине
“Информационная безопасность”

Вариант 4

Выполнил:
студент группы P34131
Бусыгин Дмитрий Алексеевич
Преподаватель:
Маркина Татьяна Анатольевна

Санкт-Петербург
2024

Цель работы

Изучение структуры и основных принципов работы современных алгоритмов блочного симметричного шифрования, приобретение навыков программной реализации блочных симметричных шифров.

По варианту необходимо реализовать шифрование алгоритмом **ГОСТ 28147-89** с режимом шифрования **CFB**

Выполнение

Основная программа

Main.kt

```
import base.decodedFileName
import base.encodedFileName
import base.inputFileName
import java.io.File
import java.io.FileNotFoundException
import java.nio.ByteBuffer
import java.nio.file.Files
import java.nio.file.Paths
import kotlin.system.exitProcess

val S_BOX = arrayOf(
    intArrayOf(0x4, 0x2, 0xF, 0x5, 0x9, 0x1, 0x3, 0xD, 0xA, 0xC, 0xB, 0x6, 0x8, 0x0, 0xE, 0x7),
    intArrayOf(0xA, 0x4, 0xD, 0x1, 0x7, 0x5, 0x0, 0xF, 0xE, 0xC, 0xB, 0x9, 0x2, 0x3, 0x6, 0x8),
    intArrayOf(0x3, 0x5, 0xD, 0xC, 0x9, 0x7, 0x1, 0xE, 0x4, 0xA, 0x0, 0xF, 0x8, 0x6, 0xB, 0x2),
    intArrayOf(0x7, 0x6, 0x4, 0x0, 0x9, 0x2, 0xB, 0xE, 0x3, 0xC, 0xD, 0xF, 0x8, 0x1, 0x5, 0xA),
    intArrayOf(0x5, 0xD, 0x6, 0xF, 0x9, 0x2, 0xC, 0xA, 0xB, 0x7, 0x8, 0x0, 0xE, 0x1, 0x3, 0x4),
    intArrayOf(0xD, 0xA, 0x7, 0xC, 0x0, 0x5, 0x4, 0xE, 0x9, 0x1, 0xB, 0x8, 0x6, 0x2, 0x3, 0xF),
    intArrayOf(0x8, 0xE, 0x2, 0x5, 0xC, 0x9, 0xA, 0x0, 0x3, 0xD, 0x7, 0xF, 0x6, 0xB, 0x1, 0x4),
    intArrayOf(0xD, 0xB, 0x4, 0x1, 0x5, 0x0, 0xF, 0x6, 0xC, 0xE, 0x2, 0x8, 0xA, 0x3, 0x7, 0x9)
)

fun encryptBlock(key: IntArray, block: Int): Int {
    var n1 = block ushr 16
    var n2 = block and 0xFFFF

    for (i in 0 until 32) {
        val temp = n1
        n1 = n2 xor f(n1, key[i % 8])  // Основная функция шифрования
        n2 = temp
    }

    return (n2 shl 16) or n1  // Объединяем блок
}

fun f(value: Int, key: Int): Int {
    val s = (value + key) and 0xFFFF
```

```
    var result = 0
    for (i in 0..3) {
        result = result or (S_BOX[i][(s ushr (i * 4)) and 0xF] shl (i * 4))
    }
    return (result shl 11) or (result ushr 5)  // Циклический сдвиг
}

fun encryptCFB(key: IntArray, iv: Int, data: ByteArray): ByteArray {
    val blocks = data.size / 2
    val result = ByteArray(data.size)
    var currentIV = iv

    for (i in 0 until blocks) {
        val block = ByteBuffer.wrap(data, i * 2, 2).short.toInt()
        val encryptedIV = encryptBlock(key, currentIV)
        val encryptedBlock = block xor encryptedIV

        ByteBuffer.wrap(result, i * 2, 2).putShort(encryptedBlock.toShort())

        // CFB
        currentIV = (currentIV and 0xFFFF0000.toInt()) or (encryptedBlock and 0xFFFF)
    }

    return result
}

fun decryptCFB(key: IntArray, iv: Int, encryptedData: ByteArray): ByteArray {
    val blocks = encryptedData.size / 2
    val result = ByteArray(encryptedData.size)
    var currentIV = iv

    for (i in 0 until blocks) {
        val encryptedBlock = ByteBuffer.wrap(encryptedData, i * 2, 2).short.toInt()
        val encryptedIV = encryptBlock(key, currentIV)

        val decryptedBlock = encryptedBlock xor encryptedIV
        ByteBuffer.wrap(result, i * 2, 2).putShort(decryptedBlock.toShort())

        // CFB
        currentIV = (currentIV and 0xFFFF0000.toInt()) or (encryptedBlock and 0xFFFF)
    }

    return result
}

fun main() {
    val key = intArrayOf(0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08)
    val iv = 0x01020304

    val line = readln()

    when (line) {
```

```
        "encode" -> {
            val message =
readTextFromFile(inputFileName).joinToString("\n").toByteArray(Charsets.UTF_16LE)

            val encodedFilePath = Paths.get(encodedFileName)
            Files.write(encodedFilePath, encryptCFB(key, iv, message))
        }
        "decode" -> {
            val message = Files.readAllBytes(Paths.get(encodedFileName))

            val decryptedFilePath = Paths.get(decodedFileName)
            Files.write(decryptedFilePath, decryptCFB(key, iv, message))
        }
        else -> {
            println("Неизвестная команда, ничего не произошло.")
        }
    }
}

fun readTextFromFile(fileName: String): List<String> {
    val lineList = mutableListOf<String>()
    try {
        File(fileName).useLines { lines -> lines.forEach { lineList.add(it) }}
    } catch (e: FileNotFoundException) {
        println("File doesn't exist!")
        exitProcess(0)
    }
    return lineList
}

object base {
    val inputFileName = "input.txt"
    val encodedFileName = "encoded.dat"
    val decodedFileName = "decoded.txt"
}
```