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

Вариант 4

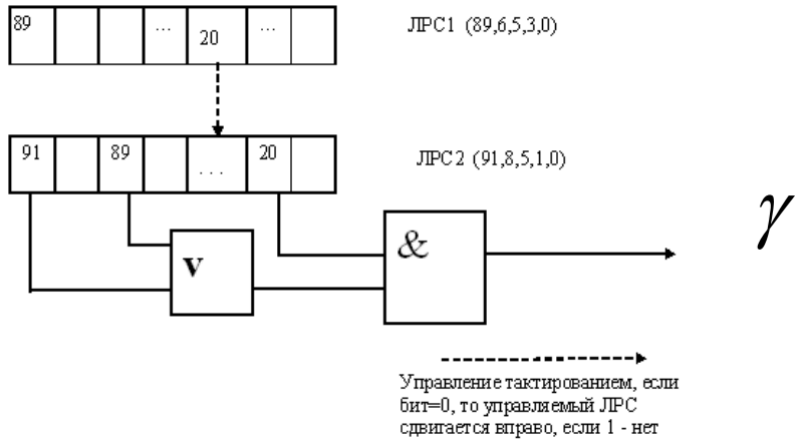
Выполнил:
студент группы P34131
Бусыгин Дмитрий Алексеевич
Преподаватель:
Маркина Татьяна Анатольевна

Санкт-Петербург
2024

Цель работы

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

По варианту необходимо реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы РС.



Выполнение

Основная программа

Main.kt

```
fun main() {
    val lfsr1 = LFSR(listOf(6, 5, 3, 0), 89)
    val lfsr2 = LFSR(listOf(8, 5, 1, 0), 91)

    println("Введите текст для кодирования:")
    val input = readLine() ?: ""

    val encodedText = codeText(input, lfsr1, lfsr2)
    println("Закодированный текст: $encodedText")

    // Сбрасываем наши ЛРС для декодирования
    lfsr1.reset()
    lfsr2.reset()

    val decodedText = codeText(encodedText, lfsr1, lfsr2)
    println("Декодированный текст: $decodedText")
}

fun codeText(text: String, lfsr1: LFSR, lfsr2: LFSR): String {
    val result = StringBuilder()

    for (char in text) {
```

```
        var encodedChar = char.toInt()
        for (i in 0 until 8) {
            // Получаем следующий бит из каждого ЛРС
            val bit1 = lfsr1.nextBit()
            val bit2 = lfsr2.nextBit()

            val keyBit = bit1 xor bit2

            val bit = (encodedChar shr i) and 1
            val encryptedBit = bit xor keyBit

            // Собираем новый закодированный символ
            encodedChar = (encodedChar and (1 shl i).inv()) or (encryptedBit shl i)
        }
        result.append(encodedChar.toChar())
    }

    return result.toString()
}

class LFSR(private val taps: List<Int>, private val length: Int) {
    private var state = (1 shl (length - 1)) - 1
    private val initialState = state

    fun nextBit(): Int {
        var newBit = 0
        for (tap in taps) {
            newBit = newBit xor ((state shr tap) and 1)
        }
        state = (state shr 1) or (newBit shl (length - 1))
        return newBit
    }

    fun reset() {
        state = initialState
    }
}
```