Национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной Инженерии и Компьютерной Техники

Вариант №6

Лабораторная работа №1.1

«Основы шифрования данных»

По дисциплине:

«Информационная безопасность»

Работу выполнила:

Студентка группы P33102

Никонова Наталья Игоревна

Преподаватель:

Рыбаков Степан Дмитриевич

Санкт-Петербург

2023

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Вариант

6. Реализовать в программе шифрование и дешифрацию файла методом биграмм с двойным квадратом. Квадраты генерировать динамически для каждого шифрования.

Листинг разработанной программы

```
abstract class RectanglePair(
   val width: Int,
   val height: Int,
   val letterString: String
) {
   private var leftMatrix: Array<Array<Char>>
   private var rightMatrix: Array<Array<Char>>
   private var letterSet: Set<Char>
   private val leftCache = mutableMapOf<Char, Pair<Int, Int>>()
   private val rightCache = mutableMapOf<Char, Pair<Int, Int>>()

   init {
      if (width <= 0)
         throw IllegalArgumentException("Can't use square with non positive square")
      if (height <= 0)
         throw IllegalArgumentException("Can't use square with non positive height")
      if (width * height != letterString.length)
         throw IllegalArgumentException("Can't use square with non equal letters and cells")

      val alreadyUsed = mutableSetOf<Char>()
      letterSet = letterString.toCharArray().toSet()
      leftMatrix = Array(height) { y ->
         Array(width) { x ->
            letterSet.minus(alreadyUsed).random().also {
               alreadyUsed.add(it)
               leftCache[it] = Pair(y, x)
            }
         }
      }
      alreadyUsed.clear()
      rightMatrix = Array(height) { y ->
         Array(width) { x ->
            letterSet.minus(alreadyUsed).random().also {
               alreadyUsed.add(it)
```

```
                rightCache[it] = Pair(y, x)
            }
        }
    }
}

fun encode(leftLetter: Char, rightLetter: Char): Pair<Char, Char> {
    return mapCoords(leftLetter, rightLetter) { oldX -> (oldX + 1) % width }
}

fun decode(leftLetter: Char, rightLetter: Char): Pair<Char, Char> {
    return mapCoords(leftLetter, rightLetter) { oldX -> (oldX - 1 + width) % width }
}

private fun mapCoords(leftLetter: Char, rightLetter: Char, newXifEqual: (Int) -> (Int)):
Pair<Char, Char> {
    val leftCoords = leftCache[leftLetter]!!
    val rightCoords = rightCache[rightLetter]!!
    val newLeftLetter: Char
    val newRightLetter: Char
    if (leftCoords.first == rightCoords.first && leftCoords.second == rightCoords.second) {
        val newX = newXifEqual(leftCoords.second)
        newLeftLetter = leftMatrix[rightCoords.first][newX]
        newRightLetter = rightMatrix[rightCoords.first][newX]
    } else if (leftCoords.first == rightCoords.first) {
        newLeftLetter = leftMatrix[leftCoords.first][rightCoords.second]
        newRightLetter = rightMatrix[rightCoords.first][leftCoords.second]
    } else {
        newLeftLetter = leftMatrix[rightCoords.first][leftCoords.second]
        newRightLetter = rightMatrix[leftCoords.first][rightCoords.second]
    }
    return Pair(newLeftLetter, newRightLetter)
}

override fun toString(): String {
    val result = StringBuilder()
    for (i in leftMatrix.indices) {
        val left = leftMatrix[i].joinToString(separator = " ")
        val right = rightMatrix[i].joinToString(separator = " ")
        result.append("$left\t\t$right\n")
    }
    return result.toString()
}

fun containsLetter(letter: Char) = letterSet.contains(letter)
```

```
    fun getAdditionalLetter() = letterString.last()
}


class RussianSquarePair : RectanglePair(
    width = 6,
    height = 6,
    letterString = "абвгдеёжзийклмнопрстуфхчцшщъыьэюя., "
)


private fun mapString(input: String, pair: RectanglePair, func: (Char, Char) -> Pair<Char,
Char>): String {
    val result = mutableListOf<Char>()
    var i = 0
    while (i < input.length) {
        if (!pair.containsLetter(input[i])) {
            result.add(input[i])
            i -= 1
        } else {
            val firstLetter = input[i]
            var secondLetter: Char
            var buffer = mutableListOf<Char>()
            if (i + 1 == input.length) {
                secondLetter = pair.getAdditionalLetter()
            } else {
                secondLetter = input[i + 1]
                while (!pair.containsLetter(secondLetter)) {
                    buffer.add(secondLetter)
                    i++
                    secondLetter = if (i + 1 == input.length) {
                        pair.getAdditionalLetter()
                    } else {
                        input[i + 1]
                    }
                }
            }
            val resultPair = func(firstLetter, secondLetter)
            result.add(resultPair.first)
            if (buffer.isNotEmpty()) result.addAll(buffer)
            result.add(resultPair.second)
        }
        i += 2
    }
    return result.joinToString(separator = "")
}


fun encode(input: String, pair: RectanglePair): String {
```

```
    return mapString(input, pair) { leftLetter, rightLetter ->
        pair.encode(leftLetter, rightLetter)
    }
}

fun decode(input: String, pair: RectanglePair): String {
    return mapString(input, pair) { leftLetter, rightLetter ->
        pair.decode(leftLetter, rightLetter)
    }
}

fun main() {
    println("Введите текст для зашифровки в одну строку:")
    var input = readlnOrNull()
    while (input == null) {
        println("Вы ничего не ввели")
        input = readlnOrNull()
    }
    input = input.lowercase(Locale.getDefault())
    println("Для удобства привели текст к нижнему регистру:\n$input")
    val pair = RussianSquarePair()
    println("Используемые матрицы для шфирования:\n$pair")
    val resultEncode = encode(input, pair)
    println("Результат шифрования:\n$resultEncode")
    val resultDecode = decode(resultEncode, pair)
    println("Результат дешифрования:\n$resultDecode")
}
```

Результаты работы

1.

```
Введите текст для зашифровки в одну строку:
Для начала займемся расстановкой точек над некоторыми фундаментальными i.
Для удобства привели текст к нижнему регистру:
для начала займемся расстановкой точек над некоторыми фундаментальными i.
Используемые матрицы для шфирования:
т ё ш ь , д           и т ж б р
п й б э з           . п ь ф э н
ч е ф щ р и         ё с а л в я
л ц ы . о я         у ч , д ю м
с в н ю м ъ         ъ ш щ е к ц
ж г у х к а         з г х о й ы

Результат шифрования:
иждуфщфёч,а.кыюкршдуфвчшчтуемшкюдьцюввп.яоп.вв,,,мя,дёыёыеяыйячтп,уц,шпін
```

```
Результат дешифрования:

для начала займемся расстановкой точек над некоторыми фундаментальными i.
```

2.

```
Введите текст для зашифровки в одну строку:

Законы природы этого Мира не только допускают, но даже провоцируют развитие так называемых «паранормальных» способностей у всего населения.

Для удобства привели текст к нижнему регистру:

законы природы этого мира не только допускают, но даже провоцируют развитие так называемых «паранормальных» способностей у всего населения.

Используемые матрицы для шфирования:

п ю ж щ н ш         в ъ х р у

я б , ь з л         ё . б с о ,

х т е д а и         з л д к э т

ы ч ц у г й         п е ю н ж г

э м ф в р           а й я щ ш м

с о к ё ъ .         ч ц ы ь ф и


Результат шифрования:

рё,фъхйаъмзшёдишбэзжэашкнтгъимтцдсюиьэнвхьгдбтйщюивзцъйазшьшкгнштшщхэёмотжтмзжирвъбэщфтпюй«анкгкюьэй.съхп»тычбьбызухифлшшшаыцзжйщзктдцк дши

Результат дешифрования:

законы природы этого мира не только допускают, но даже провоцируют развитие так называемых «паранормальных» способностей у всего населения.
```

Заключение

В ходе выполнения лабораторной работы я познакомилась с шифрованием методом биграмм с двойными квадратами. Из-за использования биграмм этот метод менее восприимчив частотному анализу и взлому, так как для этого надо проанализировать всевозможные пары букв и символов, использующихся в ключе – в случае моей программы это 1296 пар. Но, тем не менее, абсолютно не защищен.