МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №2

“Основы шифрования данных”

по дисциплине

“Информационная безопасность”

Вариант №3в

Студент:

Миху Вадим Дмитриевич

Группа P34301

Преподаватель:

Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2024

Цель работы:

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

Вариант 5:

Реализовать систему симметричного блочного шифрования, позволяющую шифровать и дешифровать файл на диске с использованием заданного блочного шифра в заданном режиме шифрования

Выполнение:

Для реализации данного алгоритма был разработан скрипт на Scala, который принимает от пользователя данны), выполняет шифрование/дешифрацию текста из указанного входного файла с использованием введенного ключевого слова, а затем сохраняет результат в выходной файл.

Листинг разработанной программы с комментариями:

```
@main
def main(): Unit = {
  val inputFile = "resources/input.txt"
  val encodeFile = "encode.txt"
  val key = "aboba".getBytes("UTF-8")

  val input = fromFile(inputFile)
  val plaintext = input.mkString
  input.close()

  if (key.length < 4 ) {
    println("Key length should be at least 4")
    exit
  }
  val S = new Array[Int](2 * r + 4)

  // Key expansion
  keyExpansion(key, S)

  // Encryption
  println(s"Converting $plaintext".take(100) + "...")
  val encrypted = encodeBlocksWithPcbc(plaintext.getBytes("UTF-8"), S, S)
  val encryptedStr = encrypted.map("%02x".format(_)).mkString
  println("Encrypted: " + encryptedStr.take(100) + "...")

  val writer = new PrintWriter(encodeFile)
  writer.write(encryptedStr)
  writer.close()

  // Decryption
  val decrypted = decodeBlocksWithPcbc(encrypted, S, S)
  println("Decrypted: " + new String(decrypted, "UTF-8").take(100) + "...")
}
```

```
// Constants for RC6 algorithm (w = 32 bits, r = 20 rounds)
val w: Int = 32
val r: Int = 20
val Pw: Int = 0xB7E15163 // Magic constant derived from the binary expansion
of e
val Qw: Int = 0x9E3779B9 // Magic constant derived from the binary expansion
of the golden ratio

// Key expansion function
def keyExpansion(key: Array[Byte], S: Array[Int]): Unit = {
  val c = key.length / 4
  val L = bytesToWords(key)

  // Initialize S array
  S(0) = Pw
  for (i <- 1 until S.length) {
    S(i) = S(i - 1) + Qw
  }

  var A = 0
  var B = 0
  var i = 0
  var j = 0
  val v = 3 * Math.max(c, S.length)

  for (s <- 0 until v) {
    A = rotateLeft(S(i) + A + B, 3)
    S(i) = A
    B = rotateLeft(L(j) + A + B, A + B)
    L(j) = B
    i = (i + 1) % S.length
    j = (j + 1) % c
  }
}

// Rotation left operation
def rotateLeft(x: Int, y: Int): Int = {
  (x << (y & (w - 1))) | (x >>> (w - (y & (w - 1))))
}

// Rotation right operation
def rotateRight(x: Int, y: Int): Int = {
  (x >>> (y & (w - 1))) | (x << (w - (y & (w - 1))))
}

// Encryption function
def encrypt(word: Array[Int], S: Array[Int]): Array[Byte] = {
  var A0 = word(0); var B0 = word(1)
  var C0 = word(2); var D0 = word(3)

  B0 += S(0)
  D0 += S(1)

  // Rc6 Algorithm
  for (i <- 1 to r) {
    val t = rotateLeft(B0 * (2 * B0 + 1), 5)
    val u = rotateLeft(D0 * (2 * D0 + 1), 5)
    A0 = rotateLeft(A0 ^ t, u) + S(2 * i)
    C0 = rotateLeft(C0 ^ u, t) + S(2 * i + 1)
    val tmp = A0; A0 = B0; B0 = C0; C0 = D0; D0 = tmp
  }

  A0 += S(2 * r + 2)
```

```
  C0 += S(2 * r + 3)

  wordsToBytes(Array(A0, B0, C0, D0))
}

// Decryption function
def decrypt(word: Array[Int], S: Array[Int]): Array[Byte] = {
  var A0 = word(0); var B0 = word(1)
  var C0 = word(2); var D0 = word(3)

  C0 -= S(2 * r + 3)
  A0 -= S(2 * r + 2)

  // Rc6 Algorithm
  for (i <- r to 1 by -1) {
    val tmp = D0; D0 = C0; C0 = B0; B0 = A0; A0 = tmp
    val u = rotateLeft(D0 * (2 * D0 + 1), 5)
    val t = rotateLeft(B0 * (2 * B0 + 1), 5)
    C0 = rotateRight(C0 - S(2 * i + 1), t) ^ u
    A0 = rotateRight(A0 - S(2 * i), u) ^ t
  }

  D0 -= S(1)
  B0 -= S(0)

  wordsToBytes(Array(A0, B0, C0, D0))
}
```

```
def xorWords(word1: Array[Int], word2: Array[Int]): Array[Int] = {
  word1.zip(word2).map((a, b) => a ^ b)
}

def encodeBlocksWithPcbc(plaintext: Array[Byte], S: Array[Int], initVector:
Array[Int]): Array[Byte] = {
  val words = bytesToWords(plaintext)
  val wordPacks = words.grouped(Const.BlockWordSize).toArray
  var vector = initVector
  var encryptedData = Array[Byte]()
  for (wordPack <- wordPacks) {
    // XOR the plaintext block with the previous ciphertext (or the IV for
the first block).
    val toEncrypt = xorWords(vector, wordPack)
    // Encrypt the result using a block cipher.
    val encrypted = encrypt(toEncrypt, S)
    // The output is the ciphertext block.
    encryptedData = encryptedData ++ encrypted
    // The next block's chaining depends on the XOR of the current plaintext
and ciphertext.
    vector = xorWords(bytesToWords(encrypted), wordPack)
  }
  encryptedData
}

def decodeBlocksWithPcbc(plaintext: Array[Byte], S: Array[Int], initVector:
Array[Int]): Array[Byte] = {
  val words = bytesToWords(plaintext)
  val wordPacks = words.grouped(Const.BlockWordSize).toArray
  var vector = initVector
  var encryptedData = Array[Byte]()
  for (wordPack <- wordPacks) {
    // Decrypt the ciphertext block using the block cipher.
    val decrypted = decrypt(wordPack, S)
```

```
    // XOR the decrypted result with the previous ciphertext (or the IV for
the first block).
    val toDecrypted = xorWords(vector, bytesToWords(decrypted))
    // The result is the plaintext block.
    encryptedData = encryptedData ++ wordsToBytes(toDecrypted)
    // As with encryption, propagate chaining using the XOR of the plaintext
and ciphertext.
    vector = xorWords(toDecrypted, wordPack)
  }
  encryptedData
}
```

Результаты работы программы:

Исходный текст:

```
Да мне всё равно на тебя, слушай. Какая у тебя там тачка, квартиры, яхты,
всё. Мне всё равно, там, хоть «Бэнтли», хоть «Майбах», хоть «Роллс-Ройс»,
хоть «Бугатти», хоть стометровая яхта. Мне на это всё равно, понимаешь?
Сколько ты там, кого имеешь, каких баб, каких вот этих самок шикарных или
атласных, в космос ты летишь, мне на это всё равно, понимаешь?
Я в своём познании настолько преисполнился, что я как будто бы уже сто
триллионов миллиардов лет проживаю на триллионах и триллионах таких же
планет, понимаешь, как эта Земля. Мне уже этот мир абсолютно понятен, и я
здесь ищу только одного: покоя, умиротворения и вот этой гармонии от слияния
с бесконечно вечным
```

Зашифрованный текст:

```
762b8ddd23f91705c9040dea06390a763d7e8a21eb1945f787bdebe1cc222d40d04ebd7917989
e89a4f8f2cf30a703a3b13f46ed09090c5e444b07991e177c1ccad9fade3962c15e3e9acc3fc5
96146f1ef8e6568213dd776f7f22711538a99582a7cb92ac6c4e1b32740b6af8826a98725c5c7
ba152e20faa6219238605c011cd5f6caa736f8ea35201b5ca13a415d18374b5385ba680f01606
ea17ae7df5c51817a0cb4db39c78c2db8f257b10c6375af2a6969cf7eb66ae267f3840ae72e7d
d0b1dc81a5153b068d4093b20beba37958966c65893fb21739cf532c9b838523e52c0ad875105
e3c57394cbb67438deec6166ed736e4128640f195007152b888da891416bd377faabd6f5e7300
6cbce2d9bcf163072170c1b29db7fcce7b541025faad0af2bcca939d134c4abdb36896ef0d09a
339f9ca6d5cfcbd26eaeafd27336c2c42cef604e1bfbbea4d4a0b3971afc8b04d8bf6c230ccb8
09a1dac1c41cc4bb7caee5ef296e4242113238bccbdc3c4e26bb9429e8d07a5c9109812e13da5
0972a0a1d6ec71b08ad9747abc183ed3527d32049624ecd8daef0767ca967695b8a0b1ccb1e36
cfec03c9b16f890e5aa1a8e8ad33eb816bace25c0ff947da5a2f68cd8d5c7733938f6bbea19ad
0133079c12b5b015b2cab59d9f07b0e3f5c9f1502b4a8c9297f73f6d77b7cb5fe09c8b22721f4
ccdaab787d783d76c4ba0203bd00246abf9bf8d24545464099284a340902b336d76978cbfbf43
2cdd80925fa700fe06691c7cf5acac4663fc4850269890f70dd7de490ad4b3b44d8eff3ef7827
69648b698787f4a209304dbc6ab72cc5fa44674f614968490176eba344912397a5d39b1a80f3a
f9875097a04ba64f4d597b772a29e63ac3173b8a9256410c3ad61dddd0499b9f3cb19f91526cc
53dc7449e45eeeae2c21fb8954e651b6e53c06283a2ab84ba333460a86a639b737ce854b85f2d
e3de3c12074f8d3060c7f113b0245bf68344f660aaa39ca6e7b959b56da0af34492ce889c4fcf
a0e764294977baa9aed59cf325fe8b0b5e5a7f3d7d6ab7c5d774f0e7e1b2163115a28c59f7f5c
98de1be6d674cf8dfc0edf22aec820eedc3f4ddea376dbe5b512233b57bc8398294452ef601d8
ea5c9368d30627b603f99e78503dfddec830de92f1ac1ecc55d2e465519d8653c65e4f38fd1c8
62ed3385ecf9b9faed326247eb29988ec1fb375402180893f4d2677d8ec86df0279fa8bcde8f7
a9bf07c9d12ceccfe7e087f36ef722a1c0f4058863225abe77c9e123ca814dd050121875ecd6c
f9580385f808cdc00dc5954844e2336b7531d1532631e01954b5960e79758c4990524ddd254b8
3ce727e9ec7d6c72f739a97dfec91c5ebee0e9131323b999ae331d198b5ba7b838ee6cd0b6397
82662c11b0dd8b7607615f5daa20ee73bf2d4f4c9aabb961aa2c8d59e7d2dfdcafebab3f95edc
570064548043bf8de89cabcfbd4cb0d19c63d866fd500ac0dedf66c25da9a5b15a02cecf61729
852543a2fef164d7b7c28a73b768e591d38ffea700c6639923fe0af88d1b8814cc66fef5a60e3
0544d59613e5aac152a716224ec3c6526388ca285831138e0ec904959d7143dc35e71658c58f4
d4c043ddc6545fd1ad3cdb3289f242d8a31d00e54e5744a2c05437aca58426d00282e4051fb71
90301434c067e
```

Расшифрованный текст:

```
Да мне всё равно на тебя, слушай. Какая у тебя там тачка, квартиры, яхты,
всё. Мне всё равно, там, хоть «Бэнтли», хоть «Майбах», хоть «Роллс-Ройс»,
хоть «Бугатти», хоть стометровая яхта. Мне на это всё равно, понимаешь?
Сколько ты там, кого имеешь, каких баб, каких вот этих самок шикарных или
атласных, в космос ты летишь, мне на это всё равно, понимаешь?
Я в своём познании настолько преисполнился, что я как будто бы уже сто
триллионов миллиардов лет проживаю на триллионах и триллионах таких же
планет, понимаешь, как эта Земля. Мне уже этот мир абсолютно понятен, и я
здесь ищу только одного: покоя, умиротворения и вот этой гармонии от слияния
с бесконечно вечным
```

Вывод:

В результате выполнения данной лабораторной работы я ознакомился с основными принципами шифрования информации, различными алгоритмами шифрования, а также реализовал алгоритм шифрования и дешифрации текста с использованием симметричного блочного алгоритма.