МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по дисциплине «Криптографические методы защиты информации»

к практической работе на тему:

**ОСНОВЫ ТЕОРИИ ЧИСЕЛ**

**И ИХ ИСПОЛЬЗОВАНИЕ В КРИПТОГРАФИИ**

Выполнил:

студент III курса 8 группы

специальности ПОИБМС

Астровская Д.А.

Проверил:

Берников Владислав Олегович

Минск 2022

**Лабораторная работа 3**

**Основы теории чисел и их использование в криптографии**

**Цель работы**: приобретение практических навыков выполнения опера-

ций с числами для решения задач в области криптографии и разра-

ботка приложений для автоматизации этих операций.

**Практическое задание**

1. Найти все простые числа в интервале [2, n]. Значение n соответствует варианту из табл. 1.2, указанному преподавателем. Подсчитать количество простых чисел в указанном интервале.

def self.find\_prime(n, m)

return false if n < 2

counter = 0

(n...m).each do |num|

if prime?(num)

puts num

counter += 1

end

end

"Колво простых чисел: #{counter}"

end

Листинг 1. - Функция нахождения простых чисел в промежутке

2. Повторить п. 1 для интервала [m, n]. Сравнить полученные результаты с «ручными» вычислениями, используя «решето Эратосфена».

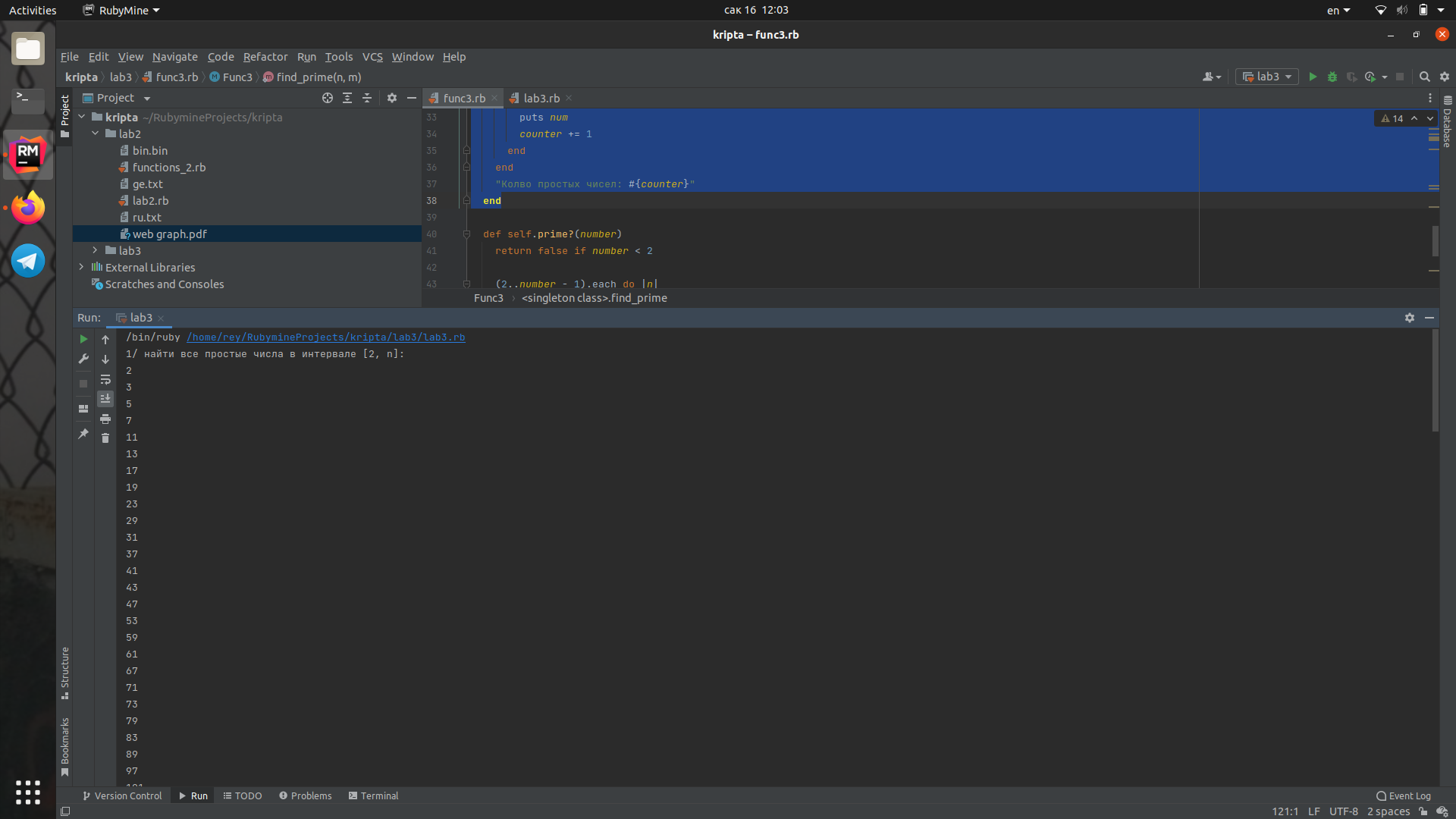


Рис.1 - Работа программы, поиск простых чисел от 2 до Н

5. Найти НОД (m, n).

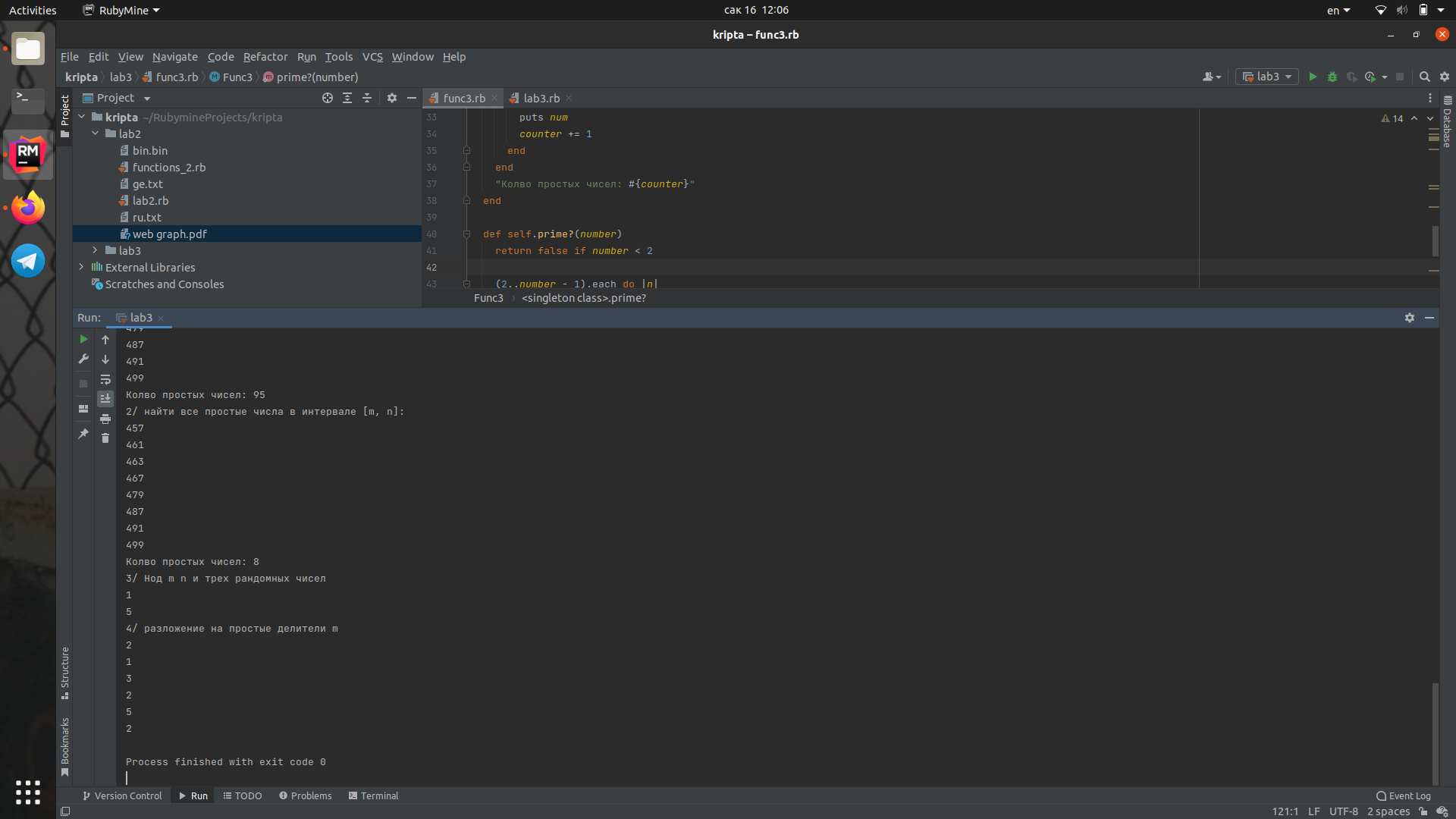


Рис.2 - Результат вычисления нода и тд

6. Разработать авторское приложение в соответствии с целью

лабораторной работы. Приложение должно реализовывать следую-

щие операции:

* вычислять НОД двух либо трех чисел;
* выполнять поиск простых чисел.

module Func3

def self.nod(a, b, c = 0)

if c < a && c < b

nod = raw\_nod(a, b)

#(c % nod).zero? ? a : 1

nod = raw\_nod(c, nod)

elsif b < a && b < c

nod = raw\_nod(a, c)

nod = raw\_nod(b, nod)

else

nod = raw\_nod(b, c)

nod = raw\_nod(a, nod)

end

nod

end

def self.raw\_nod(a, b)

until b.eql? 0

a, b = b, a % b

end

a

end

def self.find\_prime(n, m)

return false if n < 2

counter = 0

(n...m).each do |num|

if prime?(num)

puts num

counter += 1

end

end

"Колво простых чисел: #{counter}"

end

def self.prime?(number)

return false if number < 2

(2..number - 1).each do |n|

return false if (number % n).zero?

end

true

end

end

Листинг 2 - Функции для нахождения простых чисел и нода

7. С помощью созданного приложения выполнить задания по условиям.

8.Результаты выполнения работы оформить в виде отчета по установленным правилам.

**Выводы**: приобретение практических навыков выполнения опера-

ций с числами для решения задач в области криптографии и разра-

ботка приложений для автоматизации этих операций.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по дисциплине «Криптографические методы защиты информации»

к практической работе на тему:

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ**

**ШИФРОВ НА ОСНОВЕ ПОДСТАНОВКИ**

**(ЗАМЕНЫ) СИМВОЛОВ**

Выполнил:

студент III курса 8 группы

специальности ПОИБМС

Астровская Д.А.

Проверил:

Берников Владислав Олегович

Минск 2022

**Лабораторная работа 4**

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ**

**ШИФРОВ НА ОСНОВЕ ПОДСТАНОВКИ**

**(ЗАМЕНЫ) СИМВОЛОВ**

**Цель работы**: приобретение практических навыков выполнения опера-

ций с криптографическими шифрами подстановки(замены символов), различий поли- и моноалфавитных шифров.

**Практическое задание**

1. Разработать авторское приложение в соответствии с целью

лабораторной работы. Приложение должно реализовывать следую-

щие операции:

- Разработать приложение для расшифрования и зашифрования текста белорусского языка в соответствии с заданиями:

1. На основе соотношений (2.1) и (2.2) шифром Цезаря, где k = 5

@german = ' abcdefghijklmnopqrstuvwxyzäöüß'

@alpha = ' абвгдеёжзійклмнопрстуўфхцчшыьэюя'

@alpha\_length = @alpha.length

def self.alpha\_sdvig(key)

alpha\_k = @alpha

(1..key).each do |i|

alpha\_k = alpha\_k[1..-1] + alpha\_k[0, 1]

end

alpha\_k

end

def self.encrypt\_sdvigom(message, key)

encrypted = ''

puts encrypted\_alpha = alpha\_sdvig(key)

message.split('').each do |symbol|

position\_new = @alpha.index(symbol).to\_i + key.modulo(@alpha\_length)

encrypted += @alpha[position\_new]

rescue StandardError

next

end

encrypted

end

def self.decrypt\_sdvigom(message, key)

decrypted = ''

puts @alpha

message.split('').each do |symbol|

position\_new = @alpha.index(symbol) - key.modulo(@alpha\_length)

decrypted += @alpha[position\_new]

end

decrypted

end

Листинг 1 - Функции расшифрования и зашифрования для шифра цезаря

2. Виженера, ключевое слово – собственная

фамилия

def self.encrypt\_vijiner(key\_str, text\_srt)

key\_arr = str\_to\_alphabet\_index\_arr(key\_str)

char\_number\_at\_text = 0

str\_to\_alphabet\_index\_arr(text\_srt).inject("") do |r, letter\_index|

encode\_letter\_index = (letter\_index + key\_arr[char\_number\_at\_text % key\_arr.size]) % @alpha.size

char\_number\_at\_text += 1

r + @alpha[encode\_letter\_index]

end

end

def self.decrypt\_vijiner(key\_str, text\_srt)

key\_arr = str\_to\_alphabet\_index\_arr(key\_str)

char\_number\_at\_text = 0

str\_to\_alphabet\_index\_arr(text\_srt).inject("") do |r, letter\_index|

decode\_letter\_index = (letter\_index + @alpha.size - key\_arr[char\_number\_at\_text % key\_arr.size]) % @alpha.size

char\_number\_at\_text += 1

r + @alpha[decode\_letter\_index]

end

end

def self.str\_to\_alphabet\_index\_arr(str)

str.chars.map do |char|

index = @alpha.index(char)

index || raise(CryptoError, 'letters should be at alphabet')

end

end

Листинг 2- Функции расшифрования и шифрования для шифра Вижинера

Требуется рассчитать скорость работы алгоритмов, для этого установим таймеры перед началом и после окончания работы функций:

time = Time.now

text2 = 'дзе захоўваецца значная доля паступовасьці'

puts " Start message 2: #{text2}"

v = Func4.encrypt\_vijiner('астровская', text2)

puts " Encrypted message: #{v}"

time1 = Time.now

puts "#{time1 - time} sec"

puts " Decrypted message: #{Func4.decrypt\_vijiner('астровская', v)}"

time2 = Time.now

puts "#{time2 - time1} sec"

Листинг 3- Скрипт на языке Ruby для шифровки и расшифровки Вижинером с таймером выполнения

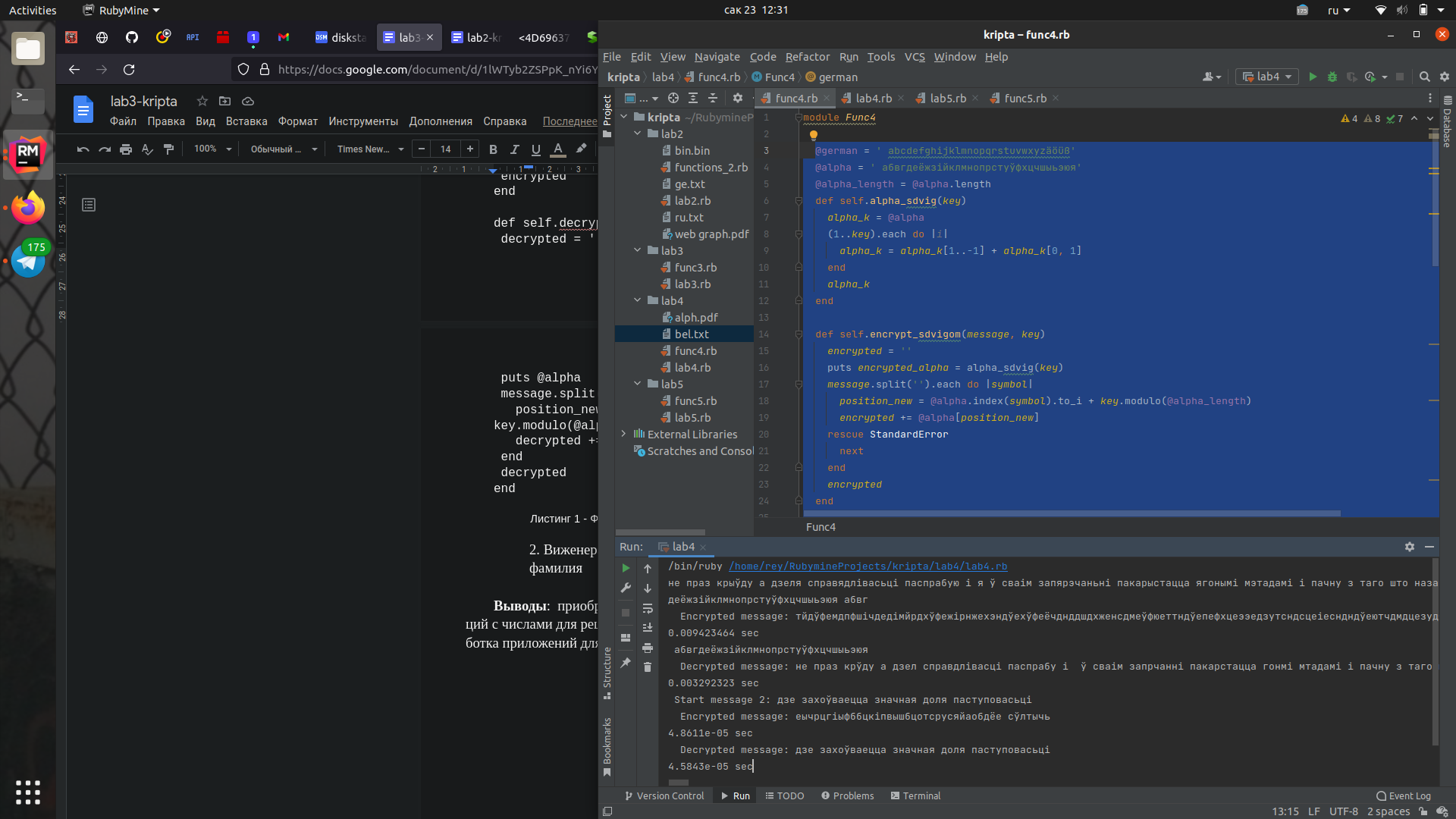


Рисунок 1- Результат работы программы

**Выводы**: приобретение практических навыков выполнения опера-

ций с моно- и поли- алфавитными перестановочными шифрами. Были получены знания о Шифрах Цезаря, Вижинера и выполнены задания шифрования и расшифровки текста на белорусском языке.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по дисциплине «Криптографические методы защиты информации»

к практической работе на тему:

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ**

**ШИФРОВ НА ОСНОВЕ**

**ПЕРЕСТАНОВКИ СИМВОЛОВ**

Выполнил:

студент III курса 8 группы

специальности ПОИБМС

Астровская Д.А.

Проверил:

Берников Владислав Олегович

Минск 2022

**Лабораторная работа 5**

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ШИФРОВ НА ОСНОВЕ ПЕРЕСТАНОВКИ СИМВОЛОВ**

**Цель работы**: приобретение практических навыков выполнения опера-

ций с числами для решения задач в области криптографии и разработка приложений для автоматизации этих операций. изучение и приобретение практических навыков разработки и использования приложений для реализации перестановочных шифров (работа рассчитана на 4 часа аудиторных занятий)

**Практическое задание**

Разработать авторское приложение в соответствии с целью

лабораторной работы. Приложение должно реализовывать следую-

щие операции:

* выполнять зашифрование/расшифрование текстовых документов (объемом не менее 500 знаков), созданных на основе алфавита языка в соответствии с нижеследующей таблицей вариантов задания; при этом следует использовать шифры подстановки из третьего столбца данной таблицы (варианты задания в табл.3.11)

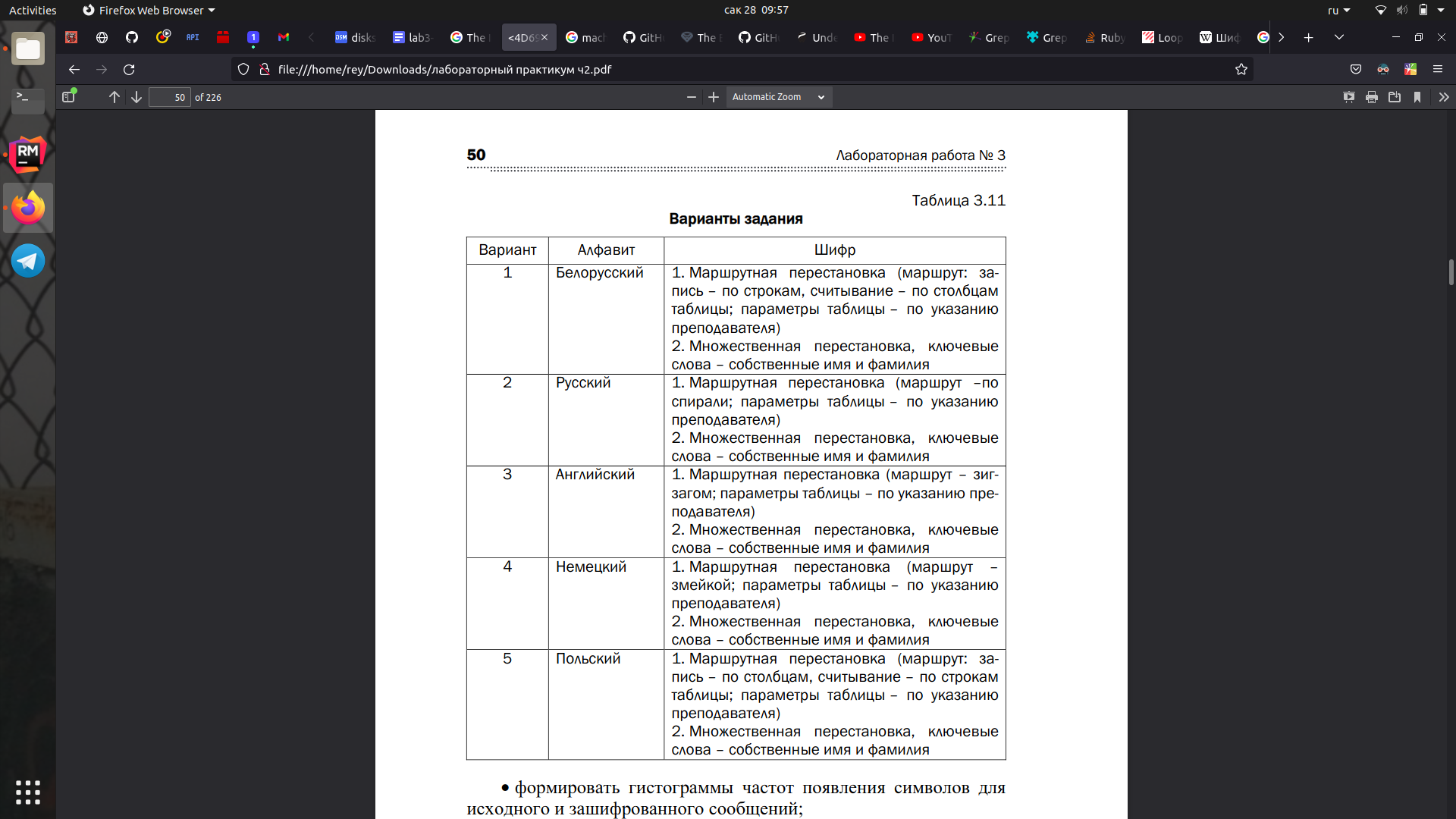


Таблица 3.11 - Вариант лабораторной работы

def self.encrypt\_skitala(message, key)

len = message.length

rows = (len.to\_f / key).ceil

m = Matrix.build(rows, key) { message.slice!(0, 1) }

puts m

result = ''

m.transpose.each do |symbol|

result += symbol

end

result

end

def self.decrypt\_skitala(message, key)

len = message.length

rows = (len.to\_f / key).ceil

m = Matrix.build(key, rows) { message.slice!(0, 1) }

# puts m

result = ''

m.transpose.each do |symbol|

result += symbol

end

result

end

Листинг 1 - Функции на языке Ruby для шифрования и расшифрования методом скитала

def self.encrypt\_change(message, name, surname)

m = Matrix.build(surname.length, name.length) { message.slice!(0, 1) }

m = m.transpose

puts m

rows = []

(0...name.length).each { |index| rows << m.row(name.index(index).to\_i).to\_a }

puts result\_temp = Matrix.rows(rows)

result\_temp = result\_temp.transpose

cols = []

(0...surname.length).each { |index| cols << result\_temp.row(surname.index(index).to\_i).to\_a }

puts result = Matrix.rows(cols)

give\_string(result)

end

def self.decrypt\_change(message, name, surname)

m = Matrix.build(surname.length, name.length) { message.slice!(0, 1) }

puts m

cols = []

surname.each { |index| cols << m.row(index).to\_a }

puts result\_temp = Matrix.rows(cols)

result\_temp = result\_temp.transpose

rows = []

name.each { |index| rows << result\_temp.row(index).to\_a }

puts result = Matrix.rows(rows)

result = result.transpose

give\_string(result)

end

Листинг 2 - Функции на языке Ruby для шифрования и расшифрования методом множественных перестановок

* формировать гистограммы частот появления символов для исходного и зашифрованного сообщений;

def self.order\_of\_letters(text)

sorted = text.chars.sort.join

result = []

text.split('').each do |symbol|

result << sorted.index(symbol).to\_i

end

result

end

Листинг 3 - Скрипт на языке Ruby для нахождения частоты появления символа в сообщении

* оценивать время выполнения операций зашифрования/расшифрования (напоминание: во многих языках программирования есть встроенные методы для замеров времени; при отсутствии такового в используемом языке можно воспользоваться разностью двух дат (например, в миллисекундах: время после выполнения программы – время до начала выполнения преобразования)).

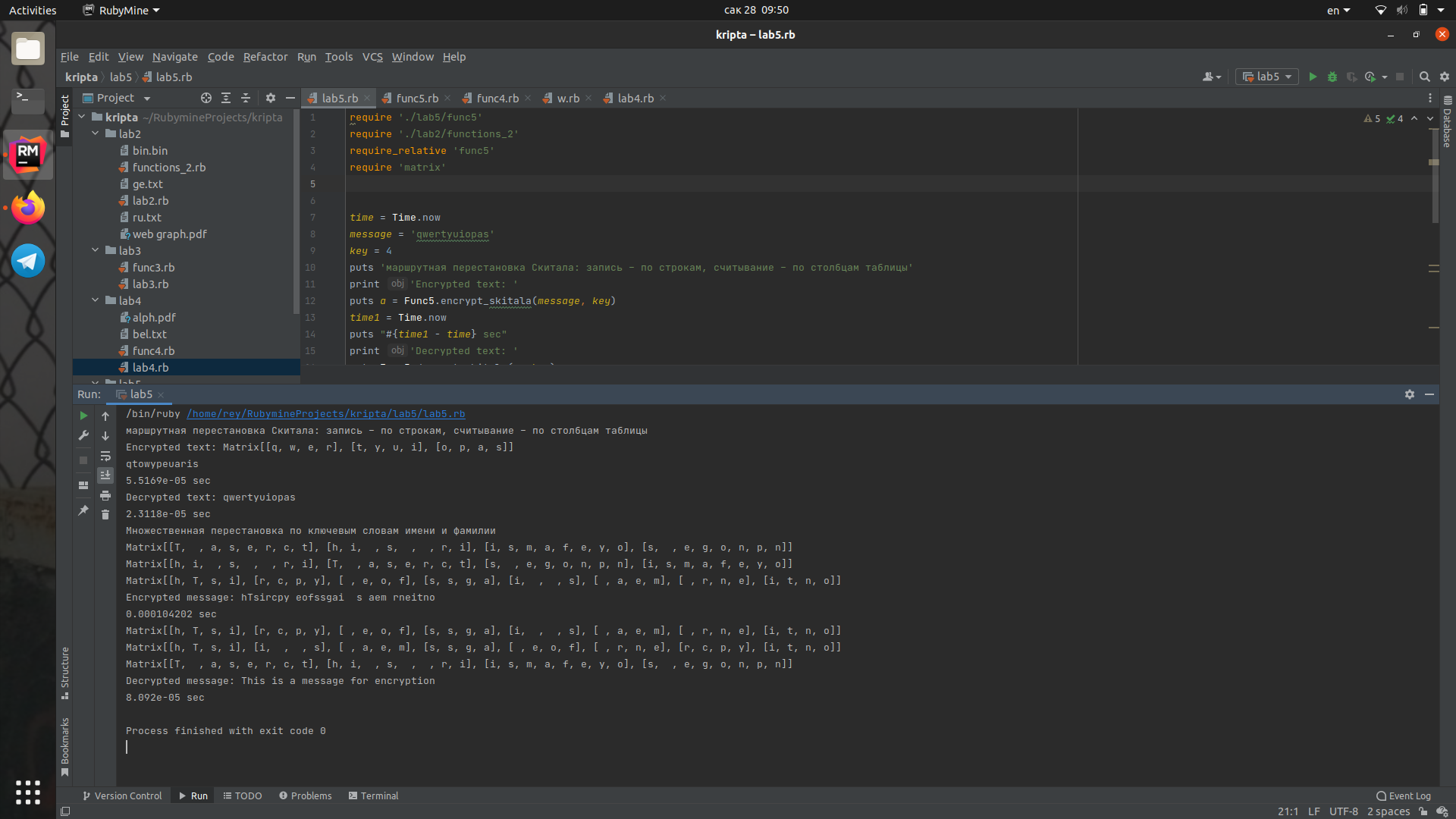


Рисунок 2- Результат работы программы с расчетом времени выполнения каждого зашифрования и расшифрования

При анализе полученных гистограмм можно сопоставить полученные данные с аналогичными результатами выполнения лабораторной работы No 2 из [2] и лабораторной работы No 2 настоящего пособия. Если указанный в таблице язык исходного текста не известен разработчику программного средства, можно взять документ на требуемом языке и воспользоваться доступным электронным переводчиком (возникающие при этом отдельные семантические неточности не следует считать существенным недостатком выполняемого анализа).

**Выводы**: приобретение практических навыков выполнения опера-

ций маршрутными перестановочными шифрами. Были получены знания о Шифрах методом скитала и выполнены задания шифрования и расшифровки текста на белорусском языке.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по дисциплине «Криптографические методы защиты информации»

к практической работе на тему:

**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**

Выполнил:

студент III курса 8 группы

специальности ПОИБМС

Астровская Д.А.

Проверил:

Берников Владислав Олегович

Минск 2022

**Лабораторная работа 7**

**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**

Цель работы: изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шиф-

ров (рассчитана на 4 часа аудиторных занятий).

**Практическое задание**

Разработать авторское приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы, из приложения в [5].

Приложение должно реализовывать следующие операции:

* разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;
* выполнение требуемых преобразований ключевой информации; выполнение операций зашифрования/расшифрования;
* оценка скорости выполнения операций зашифрования/расшифрования;
* пошаговый анализ лавинного эффекта с подсчетом количе-

ства изменяющихся символов по отношению к исходному слову.

Исследуемый метод шифрования и ключевая информация в соответствии с вариантом из табл. 5.5. Для этой лабораторный работы это алгоритм DES с ключом - Первые 8 символов собственных фамилии имени

Реализация алгоритма DES была выполнена с использованием сторонних библиотек языка Ruby. Соответствующий код расположен ниже:

require 'openssl'

class String

def encrypt(key)

cipher = OpenSSL::Cipher.new('DES').encrypt

cipher.key = key

cipher.update(self) + cipher.final

end

def decrypt(key)

cipher = OpenSSL::Cipher.new('DES').decrypt

cipher.key = key

cipher.update(self) + cipher.final

end

end

keygen = OpenSSL::Cipher.new('DES').encrypt

key = keygen.random\_key

puts "key: #{key}"

encrypted = 'Astrousk'.encrypt(key)

puts "encrypted text: #{encrypted}"

decrypted = encrypted.decrypt(key)

puts "decrypted text: #{decrypted}"

Листинг 1 - Скрипт Ruby для шифрования и расшифрования с использованием алгоритма DES

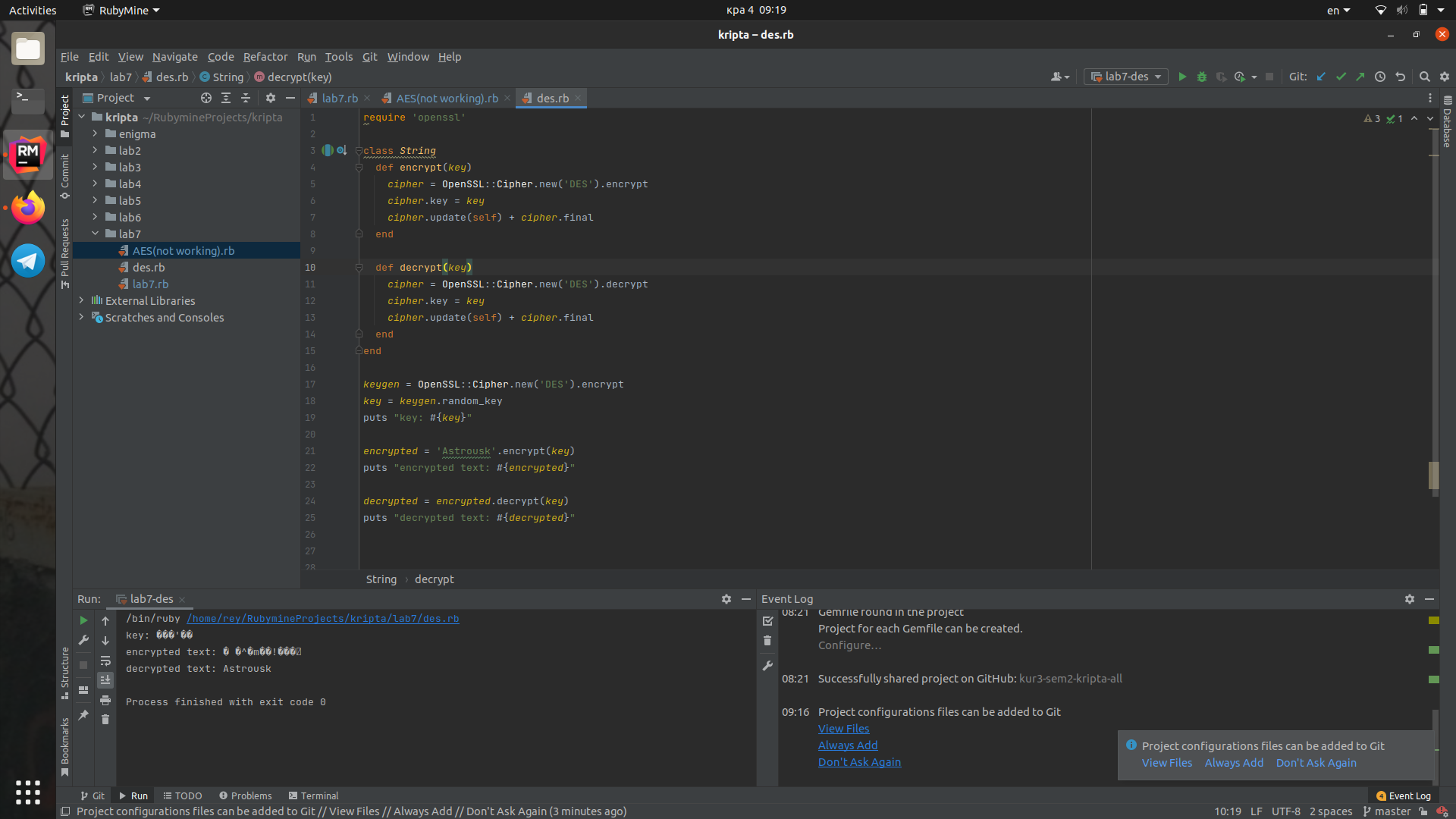


Рисунок 1- результат работы программы

m = '1234567890ABCDEF1234567890ABCDEF'.to\_bytes

k = '0f1571c947d9e8590cb7add6af7f6798'.to\_bytes # key

keys = k.expand\_key # KeyExpansion round keys are derived from the cipher key using Rijndael's key schedule

o = aes\_encrypt\_128(m, keys)

puts 'Encrypt'

puts "Out: #{o.pretty}"

o = aes\_decrypt\_128(o, keys)

puts 'Decrypt'

puts "Out: #{o.pretty}"

Листинг 2 - Скрипт на языке Ruby для использования реализованного алгоритм AES

def aes\_encrypt\_128(m, keys)

#

# Initial Round

out = m.xor\_round\_key(keys, 0) # 1. AddRoundKey�each byte of the state is combined with the round key using bitwise xor

# Rounds

# AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys

9.times do |i|

out = out.sub\_bytes # 1. SubBytes�a non-linear substitution step where each byte is replaced with another according to a lookup table.

out = out.shift\_rows # 2. ShiftRows�a transposition step where each row of the state is shifted cyclically a certain number of steps.

out = out.mix\_cols # 3. MixColumns�a mixing operation which operates on the columns of the state, combining the four bytes in each column.

out = out.xor\_round\_key(keys, i + 1) # 4. AddRoundKey

end

# Final Round (no MixColumns)

out = out.sub\_bytes # 1. SubBytes

out = out.shift\_rows # 2. ShiftRows

out.xor\_round\_key(keys, 10) # 3. AddRoundKey

end

# @param [message in bytes] m

# @param [special key in bytes] keys

def aes\_decrypt\_128(m, keys)

out = m.xor\_round\_key(keys, 10)

out = out.shift\_rows\_inv

out = out.sub\_bytes\_inv

9.times do |i|

out = out.xor\_round\_key(keys, 9 - i)

out = out.mix\_cols\_inv

out = out.shift\_rows\_inv

out = out.sub\_bytes\_inv

end

out.xor\_round\_key(keys, 0)

end

Листинг 3 - Функции на языке Ruby для шифрования и расшифрования алгоритмом AES (другие функции не включены в листинг)

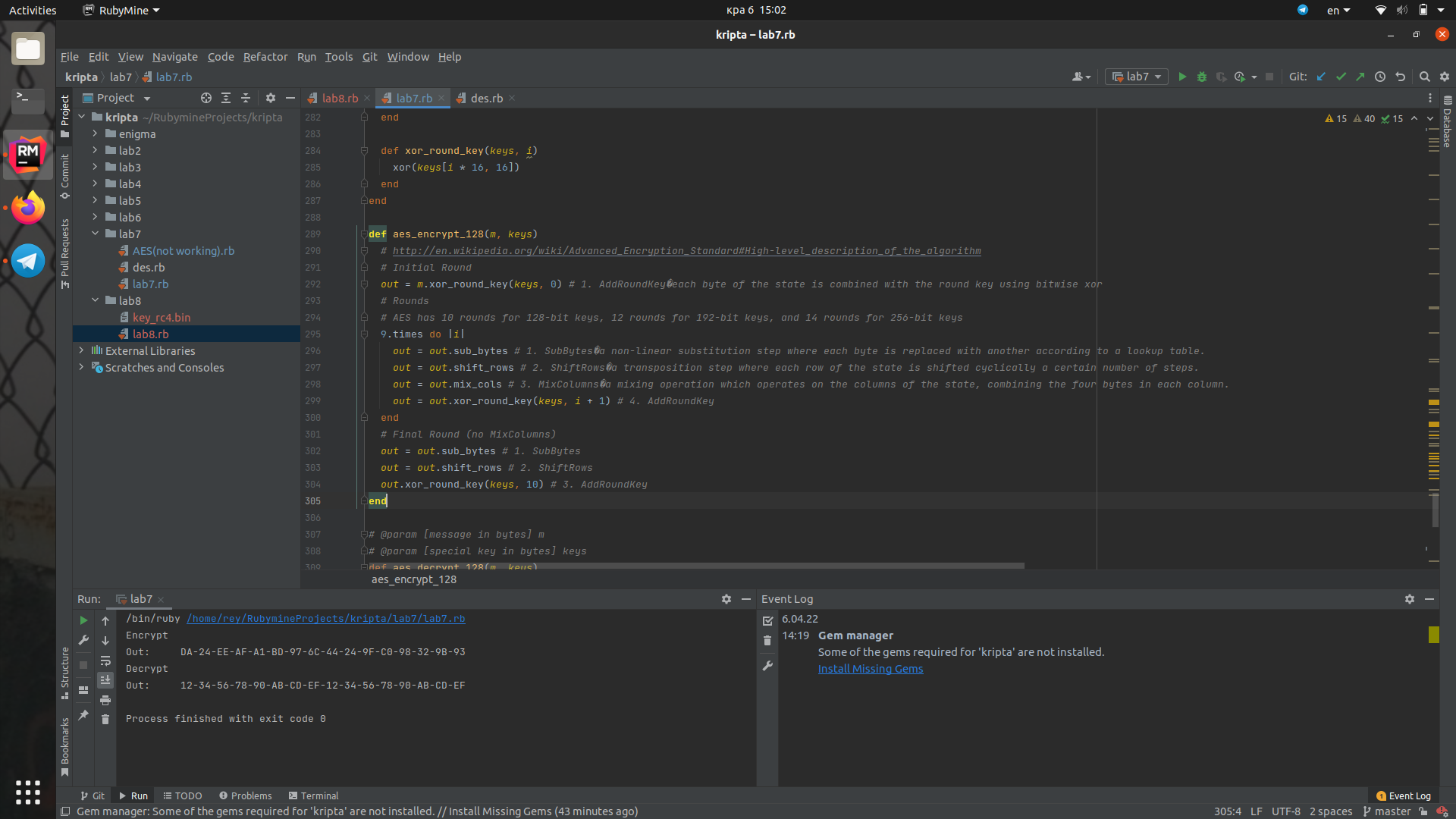


Рисунок 2- Результат работы программы

**Выводы**: Были изучены и приобретены практических навыков разработки и использования приложений для реализации блочных шифров, разработать авторское приложение в соответствии с целью лабораторной работы. При этом использовались готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы.