Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Безопасность и защита информации»

Профиль: «Компьютерные системы»

Семестр 6

ОТЧЕТ

по лабораторной работе №3

Тема: «Метод Эль-Гамаля»

Выполнил: студент группы КС-21-1б

Чирков А.В. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по использованию схемы Эль-Гамаля (Elgamal) — криптосистемы с открытым ключом, основанной на трудности вычисления дискретных логарифмов в конечном поле.

**ЗАДАНИЕ**

Реализовать шифрование текстового сообщения, используя алгоритм Эль-Гамаля.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Алгоритм шифрования Эль-Гамаля - это криптографический алгоритм, основанный на дискретном логарифмировании. Он был разработан в 1985 году Эль-Гамалем и широко используется для шифрования и подписи сообщений.

Для схемы Эль-Гамаля сначала надо получить открытый и секретный ключи:

1. Выбирается большое простое число p

2. Выбирается примитивный корень g по модулю p:

g - примитивный корень по модулю (остатку) p если:

3. Выбирается секретный ключ x, такой что

4. И вычисляется - получаем открытый ключ (p, g, h)

Шифрование:

1. Представляем блок сообщения в виде числа

2. Выбираем случайное целоев число k такое, что , и НОД(p-1, k) = 1

3. Вычисляем ; - , и есть зашифрованное сообщение

Дешифрование:

1. Вычисляем

2. Вычисляем - исходное сообщение

**ХОД РАБОТЫ**

При запуске программы, на экране появляется окно с полем ввода числа p, кнопкой Generate Key (Генерация ключей), пустыми полями ключей и кнопками шифрования (Encrypt File) и дешифрования (Decrypt File) файла (рисунок 1).

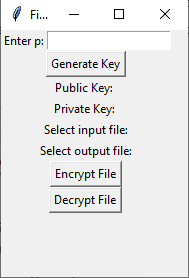


Рисунок 1 – Окно программы при запуске.

Вводим p и генерируем ключи. Пример работы программы представлен на рисунке 2.

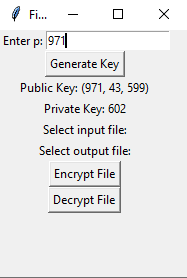


Рисунок 2 – Пример работы программы.

Нажимаем кнопку шифрования и выбираем файлы (рисунок 3).

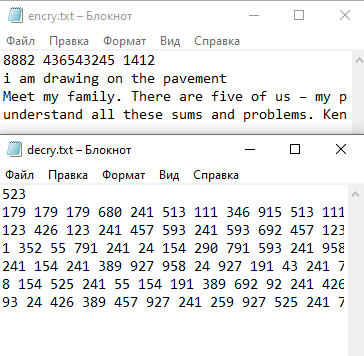


Рисунок 3 - Исходный и зашифрованный файлы

Нажимаем кнопку Decrypt и выбираем файлы для дешифрования (рисунок 4).

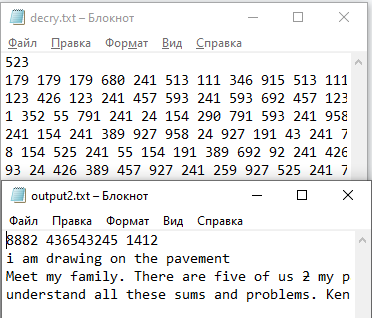


Рисунок 4 - Дешифрование файла

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

import tkinter as tk

from tkinter import filedialog

from functools import partial

import random

def is\_primitive\_root(g, p):

powers = set()

for i in range(1, p):

power = pow(g, i, p)

if power in powers:

return False

powers.add(power)

return True

def generate\_primitive\_root(p):

primitive\_roots = []

for g in range(2, p):

if is\_primitive\_root(g, p):

primitive\_roots.append(g)

return random.choice(primitive\_roots)

def generate\_keys(bits):

p = bits

g = generate\_primitive\_root(p)

x = random.randint(2, p - 2)

h = pow(g, x, p)

public\_key = (p, g, h)

private\_key = x

return public\_key, private\_key

def encrypt\_text(public\_key, text):

p, g, h = public\_key

y = random.randint(2, p - 2)

c1 = pow(g, y, p)

s = pow(h, y, p)

c2 = [(ord(char) \* s) % p for char in text]

return c1, c2

def decrypt\_text(private\_key, public\_key, ciphertext):

p, \_, \_ = public\_key

c1, c2 = ciphertext

s = pow(c1, private\_key, p)

decrypted = [chr((char \* pow(s, p-2, p)) % p) for char in c2]

return ''.join(decrypted)

def browse\_file(label):

filename = filedialog.askopenfilename()

label.config(text="Selected file: " + filename)

return filename

def encrypt\_file(public\_key, input\_file, output\_file):

with open(input\_file, 'r',encoding='utf-8') as file:

plaintext = file.read()

encrypted\_text = encrypt\_text(public\_key, plaintext)

with open(output\_file, 'w',encoding='utf-8') as file:

file.write(str(encrypted\_text[0]) + '\n')

file.write(' '.join(map(str, encrypted\_text[1])))

print("Encryption completed.")

def decrypt\_file(private\_key, public\_key, input\_file, output\_file):

with open(input\_file, 'r',encoding='utf-8') as file:

lines = file.readlines()

c1 = int(lines[0])

c2 = list(map(int, lines[1].split()))

ciphertext = (c1, c2)

decrypted\_text = decrypt\_text(private\_key, public\_key, ciphertext)

with open(output\_file, 'w',encoding='utf-8') as file:

file.write(decrypted\_text)

print("Decryption completed.")

def generate\_key\_pair(p\_entry):

global public\_key, private\_key

p = int(p\_entry.get())

public\_key, private\_key = generate\_keys(p)

public\_key\_label.config(text="Public Key: " + str(public\_key))

private\_key\_label.config(text="Private Key: " + str(private\_key))

def encrypt\_selected\_file():

global public\_key

if public\_key is None:

print("Public key not generated")

return

input\_file = browse\_file(input\_label)

output\_file = browse\_file(output\_label)

encrypt\_file(public\_key, input\_file, output\_file)

def decrypt\_selected\_file():

global private\_key

if private\_key is None:

print("Private key not generated ")

return

input\_file = browse\_file(input\_label)

output\_file = browse\_file(output\_label)

decrypt\_file(private\_key, public\_key, input\_file, output\_file)

root = tk.Tk()

root.title("File Encryption")

p\_label = tk.Label(root, text="Enter p:")

p\_label.grid(row=0, column=0)

p\_entry = tk.Entry(root)

p\_entry.grid(row=0, column=1)

generate\_button = tk.Button(root, text="Generate Key", command=partial(generate\_key\_pair, p\_entry))

generate\_button.grid(row=2, column=0, columnspan=2)

public\_key\_label = tk.Label(root, text="Public Key: ")

public\_key\_label.grid(row=3, column=0, columnspan=2)

private\_key\_label = tk.Label(root, text="Private Key: ")

private\_key\_label.grid(row=4, column=0, columnspan=2)

input\_label = tk.Label(root, text="Select input file:")

input\_label.grid(row=5, column=0, columnspan=2)

output\_label = tk.Label(root, text="Select output file:")

output\_label.grid(row=6, column=0, columnspan=2)

encrypt\_button = tk.Button(root, text="Encrypt File", command=lambda: encrypt\_selected\_file())

encrypt\_button.grid(row=7, column=0, columnspan=2)

decrypt\_button = tk.Button(root, text="Decrypt File", command=lambda: decrypt\_selected\_file())

decrypt\_button.grid(row=8, column=0, columnspan=2)

root.mainloop()