|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**Лабораторная работа №4**

**«Однонаправленные хэш-функции. Электронная цифровая подпись»**

**ДИСЦИПЛИНА: «Защита информации»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | |  |  | ( | Сафронов Н.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Ерохин И.И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

Калуга, 2023

**Цель работы:** изучить различные алгоритмы однонаправленного хэширования данных, основанные на симметричных блочных алгоритмах шифрования. Ознакомиться со схемами цифровой подписи и получить навыки создания и проверки подлинности электронной цифровой подписи.

**Постановка задачи**

1. Реализовать приложение, позволяющее вычислять и проверять ЭЦП, сформированную по алгоритмам RSA и Эль-Гамаля.
2. С помощью реализованного приложения выполнить следующие задания:
   1. Протестировать правильность работы разработанного приложения.
   2. Для заданных в варианте открытых ключей пользователя проверить подлинность подписанных по алгоритму RSA хэш-значений *m* некоторых сообщений *M*.
   3. Абоненты некоторой сети применяют подпись Эль-Гамаля с известными общими параметрами *p* и *g*. Для указанных в варианте секретных параметров абонентов найти открытый ключ и построить подпись для хэш-значения *m* некоторого сообщения *M*. Проверить правильность подписи.

**Вариант 14**

ЭЦП по алгоритму RSA:

Открытые ключи:

Проверяемые сообщения:

ЭЦП по алгоритму Эль-Гамаля:

Секретные параметры:

Хэш сообщения:

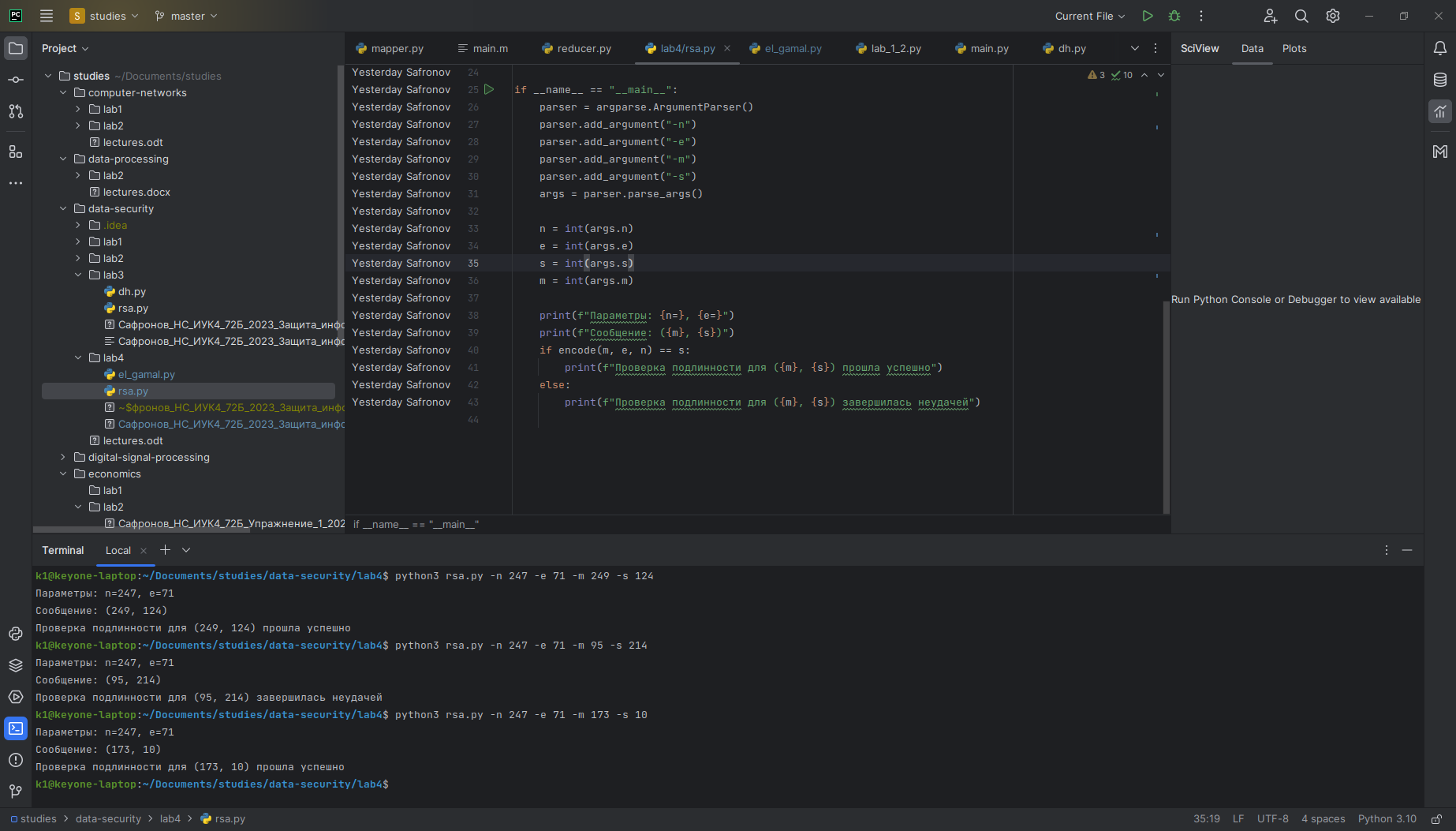
**Ход выполнения работы**

**RSA**

**Листинг программы**

import argparse  
from typing import Union  
  
  
def fast\_pow(x: int, y: int) -> Union[float, int]:  
 if y == 0:  
 return 1  
 if y == -1:  
 return 1. / x  
 p = fast\_pow(x, y // 2)  
 p \*= p  
 if y % 2:  
 p \*= x  
 return p  
  
  
def encode(message: int, e: int, n: int) -> int:  
 return fast\_pow(message, e) % n  
  
  
def decode(message: int, d: int, n: int) -> int:  
 return fast\_pow(message, d) % n  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 parser = argparse.ArgumentParser()  
 parser.add\_argument("-n")  
 parser.add\_argument("-e")  
 parser.add\_argument("-m")  
 parser.add\_argument("-s")  
 args = parser.parse\_args()  
  
 n = int(args.n)  
 e = int(args.e)  
 s = int(args.s)  
 m = int(args.m)  
  
 print(f"Параметры: {n=}, {e=}")  
 print(f"Сообщение: ({m}, {s})")  
 if encode(m, e, n) == s:  
 print(f"Проверка подлинности для ({m}, {s}) прошла успешно")  
 else:  
 print(f"Проверка подлинности для ({m}, {s}) завершилась неудачей")

**Результат выполнения программы**



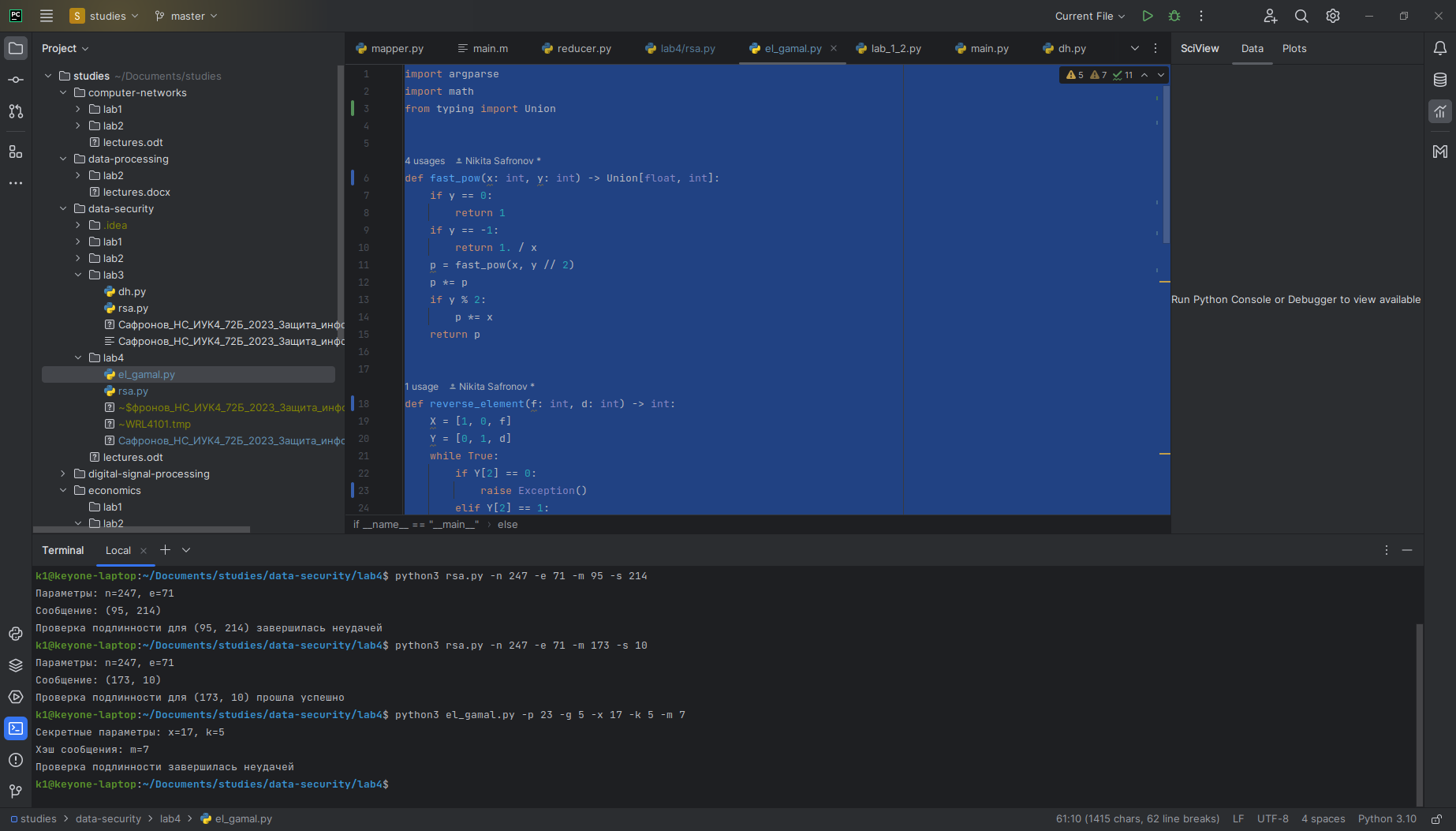
**Рисунок 1 –** Результаты работы ЭЦП по алгоритму RSA

**Эль Гамаль**

**Листинг программы**

import argparse  
import math  
from typing import Union  
  
  
def fast\_pow(x: int, y: int) -> Union[float, int]:  
 if y == 0:  
 return 1  
 if y == -1:  
 return 1. / x  
 p = fast\_pow(x, y // 2)  
 p \*= p  
 if y % 2:  
 p \*= x  
 return p  
  
  
def reverse\_element(f: int, d: int) -> int:  
 X = [1, 0, f]  
 Y = [0, 1, d]  
 while True:  
 if Y[2] == 0:  
 raise Exception()  
 elif Y[2] == 1:  
 return Y[1]  
 else:  
 q = X[2] // Y[2]  
 t = [0, 0, 0]  
 for i in range(0, len(t)):  
 t[i] = X[i] - q \* Y[i]  
 X[i] = Y[i]  
 Y[i] = t[i]  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 parser = argparse.ArgumentParser()  
 parser.add\_argument("-p")  
 parser.add\_argument("-g")  
 parser.add\_argument("-x")  
 parser.add\_argument("-k")  
 parser.add\_argument("-m")  
 args = parser.parse\_args()  
  
 p = int(args.p)  
 g = int(args.g)  
 x = int(args.x)  
 k = int(args.k)  
 m = int(args.m)  
 y = math.pow(g, x) % p  
 a = math.pow(g, k) % p  
 f = p - 1  
  
 print(f"Секретные параметры: {x=}, {k=}")  
 print(f"Хэш сообщения: {m=}")  
  
 kr = reverse\_element(f, k)  
 b = (kr \* (m - x \* a)) % f  
  
 if ((fast\_pow(y, a) \* fast\_pow(a, b)) % p) == (fast\_pow(g, m) % p):  
 print("Проверка подлинности прошла успешно")  
 else:  
 print("Проверка подлинности завершилась неудачей")

**Результат выполнения программы**

****

**Рисунок 2 –** Результаты работы ЭЦП по алгоритму Эль Гамаля

**Вывод**: в ходе выполнения лабораторной работы были изучены различные алгоритмы однонаправленного хэширования данных, основанные на симметричных блочных алгоритмах шифрования, схемы цифровой подписи, получены навыки создания и проверки подлинности электронной цифровой подписи.