Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 3

Компьютерная реализация алгоритмов криптографии с открытым ключом на примере RSA.

Выполнил: Нетецкая Ю.В.

Проверил: Олисейчик В.В.

Минск 2021

**Постановка задачи**

Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма RSA. Проверить корректность выполнения на примере.

**Описание использованных алгоритмов**

Создание ключей

|  |
| --- |
| * Выбрать простые р и q |
| * Вычислить n = p · q |
| * Выбрать e   gcd (Φ(n), e) = 1; 1 < e < Φ(n)  Число e называется открытой экспонентой. Обычно в качестве числа e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи.  Слишком малые значения e потенциально могут ослабить безопасность схемы RSA. (взаимно простое со значением функции n). |
| * Вычислить d (мультипликативно обратное к числу e по модулю n)   Число d называется секретной экспонентой. Обычно оно вычисляется при помощи расширенного алгоритма Евклида.  d = e-1 mod Φ(n) |
| * *Открытый ключ* KU = {e, n} |
| * *Закрытый ключ* KR = {d, n} |

Шифрование

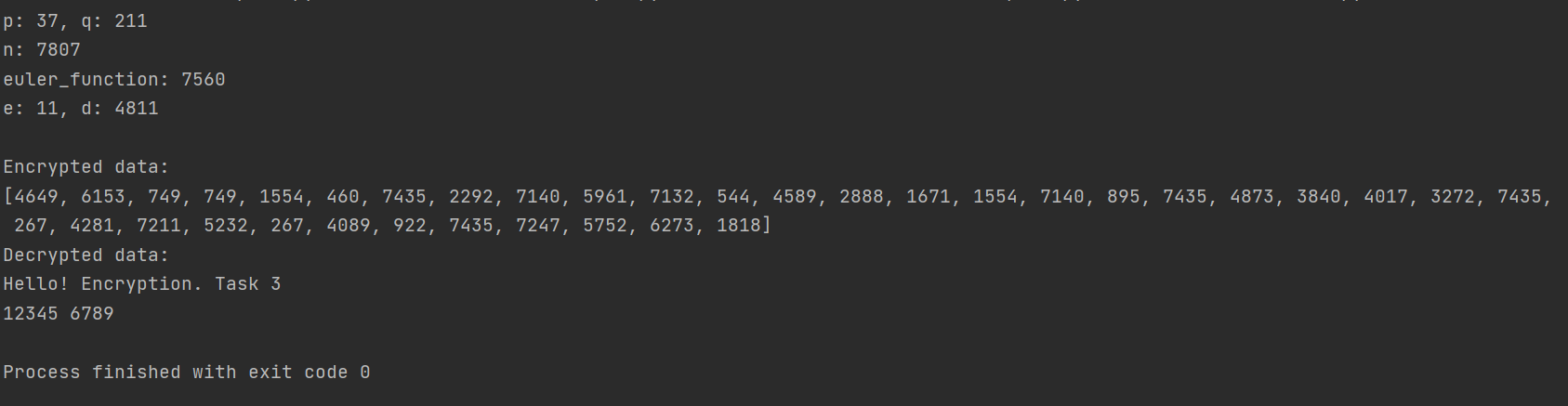
* Взять открытый ключ (e, n).
* Взять открытый текст m.

|  |
| --- |
| * Незашифрованный текст: М < n |
| * Зашифрованный текст: С = М е (mod n) |

Дешифрование

|  |
| --- |
| * Зашифрованный текст: С * Взять свой закрытый ключ (d, n). |
| * Незашифрованный текст: М = Сd (mod n) |

**Результат работы программы**



**Код программы**

import math  
import random  
from sympy import \*  
  
  
def prime\_number(n=15):  
 while True:  
 number = random.randrange(n, n\*n)  
 if isprime(number):  
 break  
 return number  
  
  
def e\_create(euler\_function):  
 for e in range(3, euler\_function, 2):  
 if math.gcd(e, euler\_function) == 1:  
 return e  
  
  
def d\_create(euler\_function, e):  
 for d in range(1, euler\_function, 1):  
 if (d \* e) % euler\_function == 1:  
 return d  
  
  
def rsa\_encrypt(input\_data, M, e, n):  
 if not (M < n):  
 print('Error')  
 return None  
 if isinstance(input\_data, str):  
 input\_data = [ord(item) for item in input\_data]  
 result\_data = []  
 for byte in input\_data:  
 C = (byte \*\* e) % n  
 result\_data.append(C)  
 return result\_data

def rsa\_decrypt(input\_data, d, n):  
 result\_data = []  
 for byte in input\_data:  
 M = (byte \*\* d) % n  
 result\_data.append(chr(M))  
 return ''.join(result\_data)  
  
  
def keys():  
 p = prime\_number()  
 q = prime\_number()  
 n = p \* q  
 euler\_function = (p - 1) \* (q - 1)  
 print('p: {}, q: {}'.format(p, q))  
 print('n: {}\neuler\_function: {}'.format(n, euler\_function))  
 return p, q, n, euler\_function  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 file = open("data.txt", "r")  
 data = file.read()  
 file.close()  
  
 p, q, n, euler\_function = keys()  
 e = e\_create(euler\_function)  
 d = d\_create(euler\_function, e)  
 print('e: {}, d: {}'.format(e, d))  
  
 encrypted\_data = rsa\_encrypt(data, euler\_function, e, n)  
 print("\nEncrypted data:\n" + str(encrypted\_data))  
  
 decrypted\_data = rsa\_decrypt(encrypted\_data, d, n)  
 print("Decrypted data:\n" + str(decrypted\_data))

**Вывод**

В данной лабораторной работе было реализовано шифрование RSA. Корректность работы алгоритмов было проверено на примере.