Министерство науки И ВЫСШЕГО ОБРАЗОВАНИЯ   
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Национальный исследовательский Томский политехнический Университет»



Инженерная школа автоматизации и робототехники

Отделение автоматизации и робототехники

15.04.04 «Автоматизация технологических процессов и производств»

Вариант 1

Лабораторная работа №4

**«Ассиметричные алгоритмы шифрования данных. Алгоритм RSA»**

по дисциплине:

**«Информационная безопасность автоматизированных систем»**

**Исполнитель:**

студент группы 8ТМ22 Гао Аозе \_\_\_\_\_\_\_\_\_\_

**Руководитель:**

К.т.н., доцент ОАР Суходоев Михаил Сергеевич \_\_\_\_\_\_\_\_\_\_

Томск – 2023

## Цель работы

Изучить особенности ассиметричных алгоритмов шифрования (алгоритмов с открытым ключом) и программно реализовать упрощенную модель алгоритма шифрования RSA.

## Ал­го­ритм RSA

Не­смот­ря на до­воль­но боль­шое чис­ло раз­лич­ных сис­те­м с от­кры­тым клю­чом, наиболее популярна – криптосистема RSA, разработанная в 1977 году и по­лу­чив­шая на­зва­ние в честь ее соз­да­те­лей: Рона Ри­ве­ста, Ади Ша­ми­ра и Леонарда Эй­дель­ма­на.

Они вос­поль­зо­ва­лись тем фак­том, что на­хо­ж­де­ние боль­ших про­стых чи­сел в вы­чис­ли­тель­ном от­но­ше­нии осу­ще­ст­в­ля­ет­ся лег­ко, но раз­ло­же­ние на мно­жи­те­ли про­из­ве­де­ния двух та­ких чи­сел прак­ти­че­ски не­вы­пол­ни­мо. До­ка­за­но (тео­ре­ма Ра­би­на), что рас­кры­тие шиф­ра RSA эк­ви­ва­лент­но та­ко­му раз­ло­же­нию. По­это­му для лю­бой дли­ны клю­ча мож­но дать ниж­нюю оцен­ку чис­ла опе­ра­ций для рас­кры­тия шиф­ра, а с уче­том про­из­во­ди­тель­но­сти со­вре­мен­ных ком­пь­ю­те­ров оце­нить и не­об­хо­ди­мое на это вре­мя

1. Вычисление ключей

Важным моментом в этом криптоалгоритме является создание пары ключей: открытого и закрытого. Для алгоритма RSA этап создания ключей состоит из сле­дующих операций:

1.1. Выбираются два простых различных числа p и q. Вычисляется их произ­ведение n = p · q, называемое модулем.

Под простым числом будем понимать такое число, которое делится только на 1 и на само себя.

Взаимно простыми числами будем называть такие числа, которые не имеют ни одного общего делителя, кроме единицы.

1.2. Вычисляется функция Эйлера Ф(n) = (p – 1) · (q – 1).

1.3. Выбирается произвольное число e (e < n), такое, что 1 < e < Ф(n), взаимно простое с числом Ф(n).

1.4. Вычисляется d как обратное число к e по модулю Ф(n), т.е. (e·d) mod Ф(n) = 1.

Обратными числами по модулю m называются такие числа n и n-1, для которых справедливо выражение (n · n-1) mod m = 1. Для вычисления обратных чисел по модулю обычно используется расширенный алгоритм Евклида.

1.5. Два числа (е, n) публикуются как открытый ключ.

1.6. Число d хранится в секрете – закрытый ключ есть пара (d, n), кото­рый позволит читать все послания, зашифрованные с помощью пары чисел (е, n).

2. Шифрование

Шифрование с помощью пары чисел производится следующим образом:

2.1. Отправитель разбивает своё сообщение M на блоки mi. Значение mi < n, поэтому длина блока mi в битах не больше k = [log2(n)] бит, где квадратные скобки обозначают, взятие целой части от дробного числа.

Например, если n = 21, то максимальная длина блока k = [log2(21)] = [4.39…] = 4 бита.

2.2. Подобный блок может быть интерпретирован как число из диапазона (0; 2k – 1). Для каждого такого числа mi вычисляется выражение (ci – зашифро­ванное сообщение):

ci = ((mi)e) mod n.

3. Дешифрование

Чтобы получить открытый текст, необходимо каждый блок дешифровать отдельно:

mi = ((ci)d) mod n.

## Задачи

1. Реализовать программу для шифрования и дешифрования сообщений с помощью алгоритма RSA. При этом:

1.1 Простые числа p и q должны задаваться пользователем.

1.2 После введения значений p и q должна осуществляться проверка, являются ли они простыми числами. Для этого могут быть использованы, например, перебор результатов деления на все значения от 2 до , тест Миллера-Рабина, малая теорема Ферма или любой другой способ.

1.3 Число е может водиться пользователем или подбираться программой. В любом случае необходимо организовать проверку, является ли число взаимно простым с Ф(n).

1.4 Число d вычисляется с помощью алгоритма Евклида по нахождению наибольшего общего делителя.

1.5 В текстовом сообщении каждая буква представляется в виде числа, соответствующего ее позиции в алфавите.

2. Осуществить проверку правильности работы программы. Для этого необходимо вручную зашифровать некоторое модельное сообщение в соответствии с выбранным алгоритмом и сравнить полученный результат с результатом работы программы. В качестве модельного сообщения используйте Вашу фамилию, записанную латиницей, и номер группы.

3. Оформить отчет по лабораторной работе.

Таблица 1. Варианты заданий

|  |  |  |
| --- | --- | --- |
| Вариант | номер группы | фамилия |
| 1 | 8TM22 | gao |

## Ход работы

Пользователь вводит два простых числа p и q. Программа должна проверить, являются ли два числа, введенные пользователем, простыми. Для шифрования F(n) = (p-1)(q-1) программа выбирает целое число e, которое взаимно простое с F(n). Для расшифровки используется метод Евклида для нахождения наибольшего общего делителя d. В текстовом сообщении каждая буква представлена числом, соответствующим ее порядковому номеру в алфавите.

Порядок действий:

Введить p и q для генерации ключа и проверки на наличие простых чисел.

Выберить функцию: зашифровать; расшифровать; выйти

-Шифрование: введите текст для шифрования.

Вывод зашифрованного результата. Возврат к выбору функции.

- Расшифровать: введите текст для расшифровки.

Выведить расшифрованный результат. Возврат к выбору функции.

В процессе экспериментов было обнаружено, что значение F(n) не является определенным при шифровании и расшифровке, что приведет к проблемам при расшифровке. Поэтому процедура была модифицирована таким образом, чтобы после шифрования выводился результат шифрования и значение n, а при расшифровке нужно было ввести список для расшифровки и значение n. Код реализован следующим образом:

Определение простоты целого числа

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

максимизировать общий делитель (математика)

def gcd(a, b):

while b:

a, b = b, a % b

return a

Рассчитать обратный элемент режима

def mod\_inverse(a, m):

m0, x0, x1 = m, 0, 1

while a > 1:

q = a // m

m, a = a % m, m

x0, x1 = x1 - q \* x0, x0

return x1 + m0 if x1 < 0 else x1

Генерация пар ключей RSA

def generate\_keypair(p, q):

if not (is\_prime(p) and is\_prime(q)):

raise ValueError("p and q must be primes")

n = p \* q

phi = (p - 1) \* (q - 1)

e = 65537

d = mod\_inverse(e, phi)

return ((n, e), (n, d))

def encrypt(message, public\_key):

n, e = public\_key

cipher\_text = [pow(ord(char), e, n) for char in message]

return cipher\_text, n

def decrypt(cipher\_text, private\_key, n):

\_, d = private\_key

decrypted\_message = ''.join([chr(pow(char, d, n)) for char in cipher\_text])

return decrypted\_message

Составление меню и программ

def main():

p = int(input("enter prime p: "))

q = int(input("enter prime q: "))

if not (is\_prime(p) and is\_prime(q)):

print("p and q must be primes.")

return

public\_key, private\_key = generate\_keypair(p, q)

while True:

print("\nChoose a function:")

print("1. Encrypt")

print("2. Decrypt")

print("3. Quit")

choice = input("Choose(1/2/3): ")

if choice == '1':

message = input("Enter the text to be encrypted: ")

cipher\_text, n = encrypt(message, public\_key)

print(f"Encryption results: {cipher\_text}, n={n}")

elif choice == '2':

cipher\_text = eval(input("Enter to be decrypted (as a list): "))

n = int(input("Enter n: "))

decrypted\_message = decrypt(cipher\_text, private\_key, n)

print(f"Decryption results: {decrypted\_message}")

elif choice == '3':

break

else:

print("Invalid choice.")

Тест программы

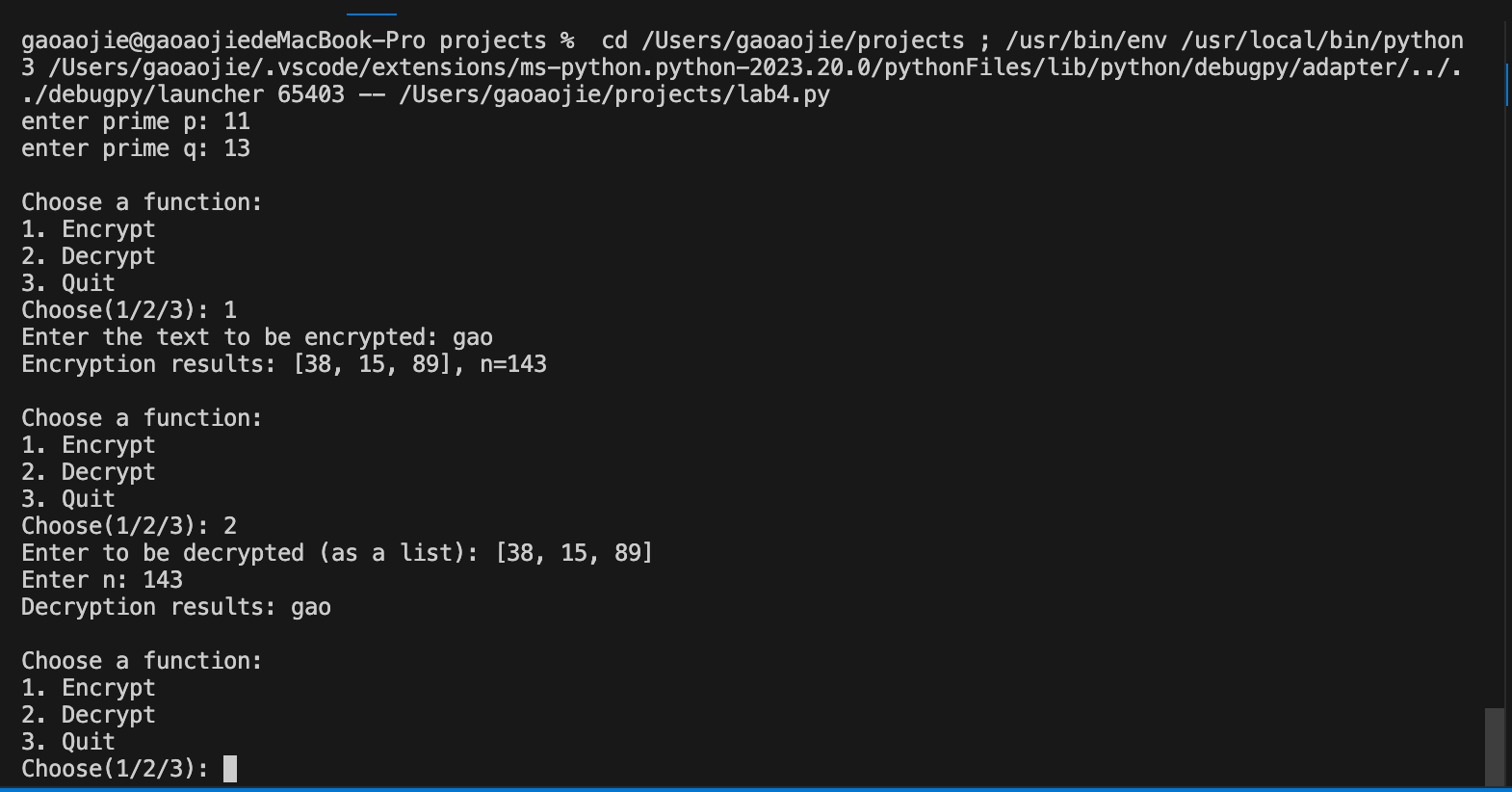


Рисунок 1 – Результат загрузки кода

Как показано на рисунке, запустить код, введите простые числа p и q, зашифровать текст "gao", результат шифрования - [38, 15, 89], при расшифровке введите список и значение n, успешно расшифровано.

Если введенные значения p и q не являются простыми числами, программа предупреждает об этом и выходит из программы.

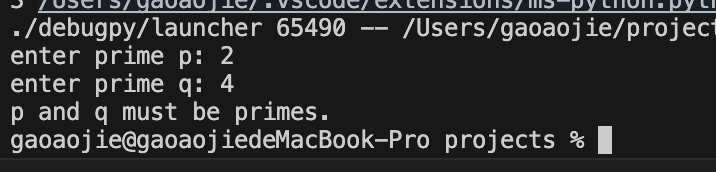


Рисунок 2 – Результат в случае вводить не простые числа

Аналогичным образом зашифровать "8TM22". Результат будет следующим:

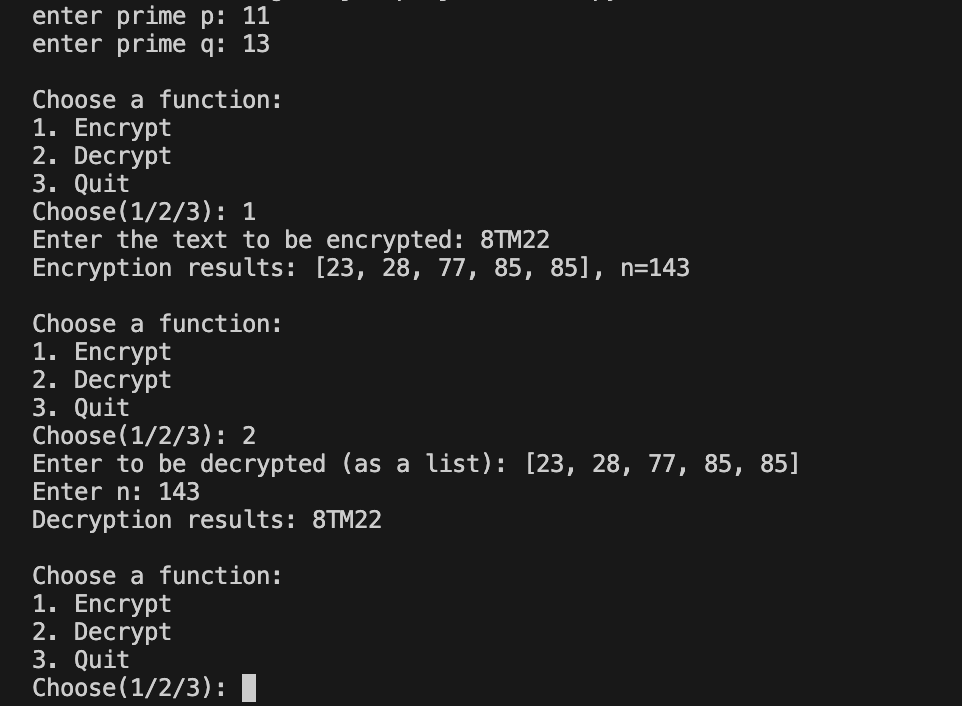


Рисунок 3 – Результат шифрование и расшифрвание "8TM22"

Проверка

enter prime p: 11

enter prime q: 13

открытый ключ：(n=143, e=7)

закрытый ключ：(n=143, d=103)

Зашифруем сообщение «САВ». Для простоты будем использовать маленькие числа (на практике применяются гораздо большие).

1. Выберем *p* = 11 и *q* = 13.
2. Определим *n* = 3\*11 = 143.
3. Найдем (*p*-1)(*q*-1) = 120. Следовательно, в качестве *е*, взаимно простое с 20, например, *е* = 65537.
4. Выберем число *d*. 113
5. Представим шифруемое сообщение как последовательность целых чисел с помощью отображения в соответствии с позициями букв в алфавите: g→103, a→97, o→111. Тогда сообщение принимает вид (103,97,111). Зашифруем сообщение с помощью ключа {7,143}.

ШТ1 = (1037) (mod 143) = 38,

ШТ2 = (977) (mod 143) = 59,

ШТ3 = (1117) (mod 143) = 45.

1. Расшифруем полученное зашифрованное сообщение (38,59,45) на основе закрытого ключа {3,33}:

ИТ1 = () (mod 143) = 103,

ИТ2= () (mod 143) = 97,

ИТ3 = () (mod 143) = 111.

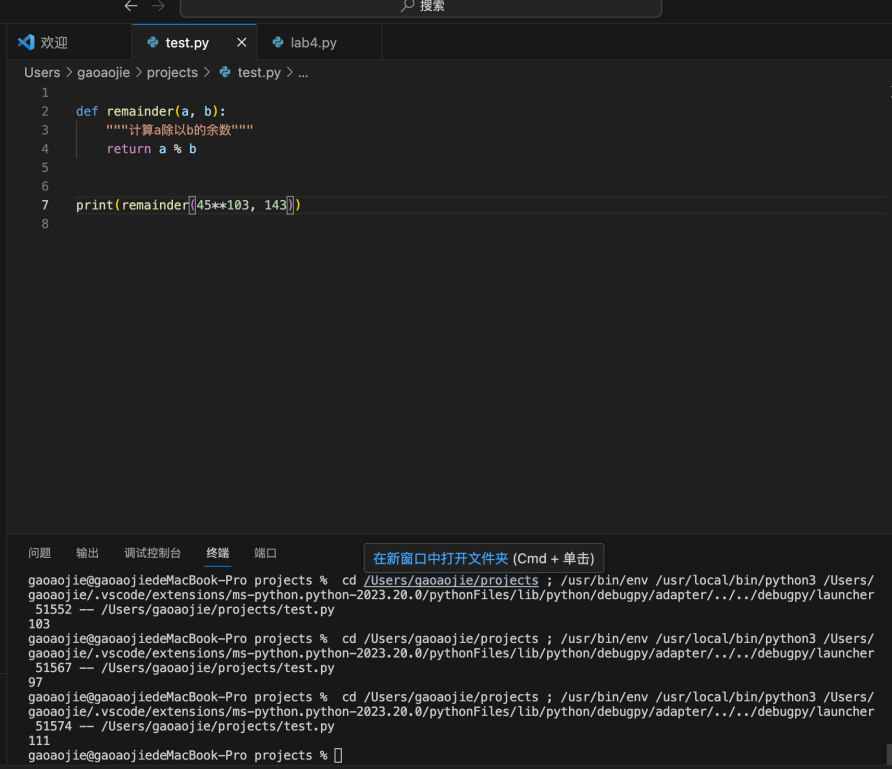


Рисунок 4 – Процедура расчета остатка

## Вывод

В этом эксперименте с помощью языка python реализован алгоритм шифрования RSA, который зашифровывает и расшифровывает имена, а также проводятся письменные вычисления для проверки правильности процедуры. Получены знания и реализация алгоритма шифрования.

## Приложение

def is\_prime(num):

"""Check if a number is prime"""

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

def gcd(a, b):

"""Calculate the greatest common divisor"""

while b:

a, b = b, a % b

return a

def mod\_inverse(a, m):

"""Calculate the mode inverse element"""

m0, x0, x1 = m, 0, 1

while a > 1:

q = a // m

m, a = a % m, m

x0, x1 = x1 - q \* x0, x0

return x1 + m0 if x1 < 0 else x1

def generate\_keypair(p, q):

"""Generate RSA key pairs"""

if not (is\_prime(p) and is\_prime(q)):

raise ValueError("p and q must be primes")

n = p \* q

phi = (p - 1) \* (q - 1)

# 选择一个与phi互质的整数e

e = 7 # 通常选择65537作为e

# 计算e的模反元素d

d = mod\_inverse(e, phi)

# 输出公钥和私钥

print(f"公钥：(n={n}, e={e})")

print(f"私钥：(n={n}, d={d})")

# 返回公钥和私钥

return ((n, e), (n, d))

def encrypt(message, public\_key):

"""加密消息"""

n, e = public\_key

cipher\_text = [pow(ord(char), e, n) for char in message]

return cipher\_text,n

def decrypt(cipher\_text, private\_key, n):

"""解密消息"""

\_, d = private\_key

decrypted\_message = ''.join([chr(pow(char, d, n)) for char in cipher\_text])

return decrypted\_message

def main():

p = int(input("enter prime p: "))

q = int(input("enter prime q: "))

if not (is\_prime(p) and is\_prime(q)):

print("p and q must be primes.")

return

public\_key, private\_key = generate\_keypair(p, q)

while True:

print("\nChoose a function:")

print("1. Encrypt")

print("2. Decrypt")

print("3. Quit")

choice = input("Choose(1/2/3): ")

if choice == '1':

message = input("Enter the text to be encrypted: ")

cipher\_text, n = encrypt(message, public\_key)

print(f"Encryption results: {cipher\_text}, n={n}")

elif choice == '2':

cipher\_text = eval(input("Enter to be decrypted (as a list): "))

n = int(input("Enter n: "))

decrypted\_message = decrypt(cipher\_text, private\_key, n)

print(f"Decryption results: {decrypted\_message}")

elif choice == '3':

break

else:

print("Invalid choice.")

if \_\_name\_\_ == "\_\_main\_\_":

main()