МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		С.Ю. Гуков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №9

Асимметричный алгоритм шифрования RSA

по курсу:

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	4329		Д.С. Шаповалова
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

Цель работы

Программно реализовать на любом языке программирования ассиметричный алгоритм шифрования RSA.

Постановка задачи

Задание: Написать программу, реализующую ассиметричный алгоритм шифрования RSA. Продемонстрировать на примерах его работу по зашифровыванию и расшифровыванию текста. Разрешается реализовать как версию с пользовательским интерфейсом, так и консольную версию с дружелюбным командно-текстовым интерфейсом. Текст задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

Текст задания
Последовательность шагов алгоритма RSA: - выбрать два больших простых числа p и q; - вычислить: n=p*q и фи(n) = (p-1)*(q-1); - выбрать случайное число e, взаимно простое с фи(n) ; - определить такое число d, для которого является истинным выражение: (e*d)mod(фи(n))=1; - числа e и n - это открытый ключ, а числа d и n - это закрытый ключ; На практике это означает следующее: открытым ключом зашифровывают сообщение, а закрытым - расшифровывают. Пара чисел закрытого ключа держится в секрете. - разбить шифруемый текст на блоки, каждый из которых может быть представлен в виде числа M (i); Обычно блок берут равным одному символу и представляют этот символ в виду числа - его номера в алфавите или кода в таблице символов (например ASCII или Unicode). - шифрование алгоритмом RSA производится по формуле: C(i) = (M(i)^e)mod n; - расшифровка сообщения производится по формуле: M(i)=(C(i)^d)mod n.

Текст задания
mod – операция взятия остатка от деления. взаимно простыми называются такие числа, которые не имеют между собой ни одного общего делителя, кроме единицы.

Схема алгоритма решения

1. Установка ключей (генерация/ввод)
2. Обработка сообщения (шифрование/дешифрование)

Полное описание реализованной функции

Функция ввода команд (дружелюбный консольный интерфейс):

1. Сгенерировать ключи
    a. Вводится длина простых чисел в битах
    b. Генерируется 2 простых числа - p и q
    c. На основе них генерируется пара ключей (ключ - пара чисел (связанных математически)) - открытый и закрытый (см. функцию генерации ключей (generate_keypair))
    d. Ключи выводятся в консоль
2. Ввести открытый ключ
    a. Вводятся числа e и n
3. Ввести закрытый ключ
    a. Вводятся числа d и n
4. Зашифровать сообщение
    a. Проверяется наличие открытого ключа
    b. Передача сообщения в функцию шифрования (encrypt)
    c. Вывод зашифрованного
5. Расшифровать сообщение
    a. Проверяется наличие закрытого ключа
    b. Передача сообщения в функцию шифрования (decrypt)
    c. Вывод расшифрованного

Функция генерации ключей (generate_keypair**):**

1. Генерируем 2 простых числа заданной длины(битности) - отдельная функция - перебираем рандомные (random.getrandbits) нечётные числа, пока не найдём простое (сверяемся с таблицей простых чисел (sympy : isprime))

2. Перемножаем p и q = n = модуль для RSA шифрования. Вычисляем функцию Эйлера = фи(n) = (p - 1) * (q - 1)
3. Выбираем такое e, которое взаимно простое для фи(n) - их НОД(gcd) = 1, обычно e = 65537 (для открытого ключа)
4. Определяем d (для закрытого ключа) - обратный элемент для e по модулю фи(n). Вычисляется по формуле - остаток от деления на n от результат возведения e в -1 степень.
5. Возвращаем пары пар чисел - (e, n), (d, n)

Функция шифрования (encrypt**):**

1. Преобразует каждый символ строки в его код ASCII (ф-я ord) => получаем массив последовательных чисел
2. Возвращаем остаток от деления на n результата возведения символа-числа в степень e = (num^e) mod n
3. Возвращаем теперь непонятные числа

Функция дешифрования (decrypt**):**

1. Перебираем числа - от каждого числа берём остаток от деления на n результата возведения числа в степень d (обратный элемент для e по модулю фи(n))
2. Получаем снова символы-числа в ASCII => с помощью ф-и chr преобразуем символ-число в символ-букву => объединяем в строку.
3. Возвращаем понятные буквы

Листинг программы

```
import random
from math import gcd
from sympy import isprime

# Генерация простого числа заданной битности
def generate_large_prime(length):

    while True:
        candidate = random.getrandbits(length) | 1  # Генерируем нечетное число
        if isprime(candidate):
            return candidate


def generate_keypair(p, q):
    # p и q - простые числа (генерация выше)
    n = p * q # модуль rsa
    # Вычисление функции Эйлера
    phi = (p - 1) * (q - 1)

    # Проверка на взаимную простоту
    def is_coprime(e, phi):
        return gcd(e, phi) == 1 # НОД

    e = 65537  # Обычно выбирается простое число 65537
    if not is_coprime(e, phi):
        raise ValueError("e не является взаимно простым с phi(n)")

    # Вычисление закрытой экспоненты d
    d = pow(e, -1, phi)

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

'''
# Вычисление обратного элемента по модулю
def modinv(a, m):
    m0, x0, x1 = m, 0, 1
    if m == 1:
        return 0
    while a > 1:
        q = a // m
        m, a = a % m, m
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += m0
    return x1
'''

# Шифрование сообщения на основе открытого ключа
def encrypt(pk, plaintext):
    numbers = [ord(char) for char in plaintext]
    ciphertext = [pow(num, pk[0], pk[1]) for num in numbers]
    return ciphertext


# Расшифровка сообщения на основе закрытого ключа
def decrypt(sk, ciphertext):
    plaintext_numbers = [pow(c, sk[0], sk[1]) for c in ciphertext]
    message = ''.join(chr(num) for num in plaintext_numbers)
    return message
```

```python
def main():
    print("Добро пожаловать в программу шифрования RSA! 😊")

    public_key, private_key = None, None

    while True:
        print("\nВыберите опцию:")
        print("1. Сгенерировать ключи")
        print("2. Ввести открытый ключ вручную")
        print("3. Ввести закрытый ключ вручную")
        print("4. Зашифровать сообщение")
        print("5. Расшифровать сообщение")
        print("6. Выход")

        choice = input("Ваш выбор: ")

        if choice == '1':
            length = int(input("Введите длину простых чисел в битах: "))
            p = generate_large_prime(length)
            q = generate_large_prime(length)
            public_key, private_key = generate_keypair(p, q)
            print("Открытый ключ:", public_key)
            print("Закрытый ключ:", private_key)

        elif choice == '2':
            e = int(input("Введите значение e: "))
            n = int(input("Введите значение n: "))
            public_key = (e, n)
            print("Открытый ключ установлен:", public_key)

        elif choice == '3':
            d = int(input("Введите значение d: "))
            n = int(input("Введите значение n: "))
            private_key = (d, n)
            print("Закрытый ключ установлен:", private_key)

        elif choice == '4':
            if public_key is None:
                print("Сначала сгенерируйте или введите открытый ключ! 🔑")
                continue
            message = input("Введите сообщение для шифрования: ")
            encrypted_message = encrypt(public_key, message)
            print("Зашифрованное сообщение:", encrypted_message)

        elif choice == '5':
            if private_key is None:
                print("Сначала сгенерируйте или введите закрытый ключ! 🔑")
                continue
            ciphertext_input = input("Введите зашифрованное сообщение (список
чисел, разделенных запятыми): ")
            ciphertext = list(map(int, ciphertext_input.split(',')))
            decrypted_message = decrypt(private_key, ciphertext)
            print("Расшифрованное сообщение:", decrypted_message)

        elif choice == '6':
            print("Выход из программы. Всего хорошего! 👋")
            break

        else:
            print("Неверный выбор, попробуйте снова.")


if __name__ == "__main__":
    main()
```

Результат выполнения программы.

```
Добро пожаловать в программу шифрования RSA! 😊

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 1
Введите длину простых чисел в битах: 8
Открытый ключ: (65537, 6931)
Закрытый ключ: (145, 6931)

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 2
Введите значение e: 65537
Введите значение n: 6931
Открытый ключ установлен: (65537, 6931)

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 4
Введите сообщение для шифрования: Карга
Зашифрованное сообщение: [3319, 2754, 2367, 6343, 2754]

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 3
Введите значение d: 145
Введите значение n: 6931
Закрытый ключ установлен: (145, 6931)

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 5
Введите зашифрованное сообщение (список чисел, разделенных запятыми): 3319, 2754, 2367, 6343,
 2754
Расшифрованное сообщение: Карга

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 6
Выход из программы. Всего хорошего! 👋
```

Рисунок 2.1 - Работа программы кратко (8 бит)

```
Добро пожаловать в программу шифрования RSA! 😊

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 1
Введите длину простых чисел в битах: 1024
Открытый ключ: (65537,
 12238962459508637271226298094176116992329541164770394886293439973214640879368680625670938668131
 74365446559479942981980132334762796238684990907795224110464670458649099447453553432797173111506
 56360577095963427550252882942254533667348564073662389759537456619087848285697366031970106453150
 11806242519439762567052054373220315835858865952469443376567996619614531879122881125628570396678
 38590412757539721891026048099304713937317289044685003035115424992364180255924749148923272638405
 08859315613635265907114720317463103010144112224620258086628516435066188276301917051033482819939
 88707681171197266314043174489091479645100480103)
Закрытый ключ:
 (43568517658778477430719980245834995411912079493125000030093833189663483485004092191724216721472
 23865081835187254066704250841051864385744248865203817362667512425046668143325013375454446323320
 38587244818176400579022533210900686184281276223179937334482946471814692471509522094784054557761
 14644241237446475436613193771455039872446044589784087100133954547900930414548148955444174336077
 00276138954009288042490838868126133828659483397982105499561371259826964730229674642293269629392
 81324562391829377909728794264422797854967515683957732001336677492441773096082050375992231737691
 48952669670110406538898708450045664168834619073,
 12238962459508637271226298094176116992329541164770394886293439973214640879368680625670938668131
 74365446559479942981980132334762796238684990907795224110464670458649099447453553432797173111506
 56360577095963427550252882942254533667348564073662389759537456619087848285697366031970106453150
 11806242519439762567052054373220315835858865952469443376567996619614531879122881125628570396678
 38590412757539721891026048099304713937317289044685003035115424992364180255924749148923272638405
 08859315613635265907114720317463103010144112224620258086628516435066188276301917051033482819939
 88707681171197266314043174489091479645100480103)
```

Рисунок 2.1 - Генерация пары ключей (1024 бит)

```
Ваш выбор: 4
Сначала сгенерируйте или введите открытый ключ! 🔑

Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 2
Введите значение e: 65537
Введите значение n:
 817238537119660837480354867284088957788276134832501708085697197776956243679223575251728634297908240250896945470500676979303911754736749764627426996725781448726373352540244645248992670476129484228481220761019322327553476329959121968940068685517819708038874509809200625988378438524098891440172874727136869579653909433685280504371533420224717336334348660180173364657943434850803002976081239693514212998186763187903169440670690870820993385164310052959288837699194234597719753207900469886362193355005521203012251907665803118659301784096105112197404981414566029844567922854252874863336627779698455588891873482232760546237
Открытый ключ установлен: (65537,
 817238537119660837480354867284088957788276134832501708085697197776956243679223575251728634297908240250896945470500676979303911754736749764627426996725781448726373352540244645248992670476129484228481220761019322327553476329959121968940068685517819708038874509809200625988378438524098891440172874727136869579653909433685280504371533420224717336334348660180173364657943434850803002976081239693514212998186763187903169440670690870820993385164310052959288837699194234597719753207900469886362193355005521203012251907665803118659301784096105112197404981414566029844567922854252874863336627779698455588891873482232760546237)
```

Рисунок 2.2 - Установка открытого ключа (1024 бит)

```
Выберите опцию:
1. Сгенерировать ключи
2. Ввести открытый ключ вручную
3. Ввести закрытый ключ вручную
4. Зашифровать сообщение
5. Расшифровать сообщение
6. Выход
Ваш выбор: 4
Введите сообщение для шифрования: Карга
Зашифрованное сообщение:
 [800887646282296105084878840087523732719831890625296987034284129982134373531323956702767387782693384327378174806702469981421726680741869674572061072701771384341548791350709012988946096753800092188152210714892488346389774463314450810365959975746582955587046028363195209028674598647456185578358134680054445176844881345658350928948456057115287118598006289861987433495361154287151526619066587575479313584709649771963679462172939148771487153616397663614466904184240228220668194452163819711592956370026007822922388943206202305675603554504377489744476526637169474103778705101414192213816913961093994725291791432214492944458,678760411382494110277605386691935591107566220597466550588231529909177833641260352648461804353953637849245255886546083344057460278106610167527462675739160493811568164378519496725354383900382849275046273043124267672075172001775051176738739669569236246532287023551190193766245509920303124914387933900717162935440425939142920422031407653011709660167156305389260797461464437080149268452876320115764148942264582448827477292114048657945931147379263596586806488799327303268598242913606145610233041648096631403343216566584035446332142852208881412657437737675869481569501508964794586093185618963688369783187858824212316245857,104458756774129461397541662897690518353726499167144658761983956379473126413894300448799595212126396515233338524425579388908736883433252590744012030924089642649584428431946449181390694215479462393490213179855945721756631526768751313923266116909491348569522069979839827801512662636470890435908394028162016907750114436595908796412449250004108597904010448186140386984061236828816122429180253007893177632382210694869296571518867733751582974646904070559330559656024587708608072041255529871525479409798064236031312468925773675838270006314496151904811595386192649954744626751703887786335554936458491426314435488657575986655,168138398534118709316029448017718776592130321401188213182016238685867336364111565024275381116916697987297307319813171759631685116253304893348694027810836119405108655426901901939262759516285543391603140389235942069742041133978324501680813180510553766687351183403774896747662276896875383697618050361450005515988288355705023971598684001565857384475477032277937737663801665801814821256327419386208299890147093785573976283933287468661298376555629649966695962222046257702669535412774359337438801785337385843636472503205784421819121443990480397420388523607348548575139541414917968476535427025247961692260582726022385734828,678760411382494110277605386691935591107566220597466550588231529909177833641260352648461804353953637849245255886546083344057460278106610167527462675739160493811568164378519496725354383900382849275046273043124267672075172001775051176738739669569236246532287023551190193766245509920303124914387933900717162935440425939142920422031407653011709660167156305389260797461464437080149268452876320115764148942264582448827477292114048657945931147379263596586806488799327303268598242913606145610233041648096631403343216566584035446332142852208881412657437737675869481569501508964794586093185618963688369783187858824212316245857]
```

Рисунок 2.3 - Шифрование сообщения (1024 бит)

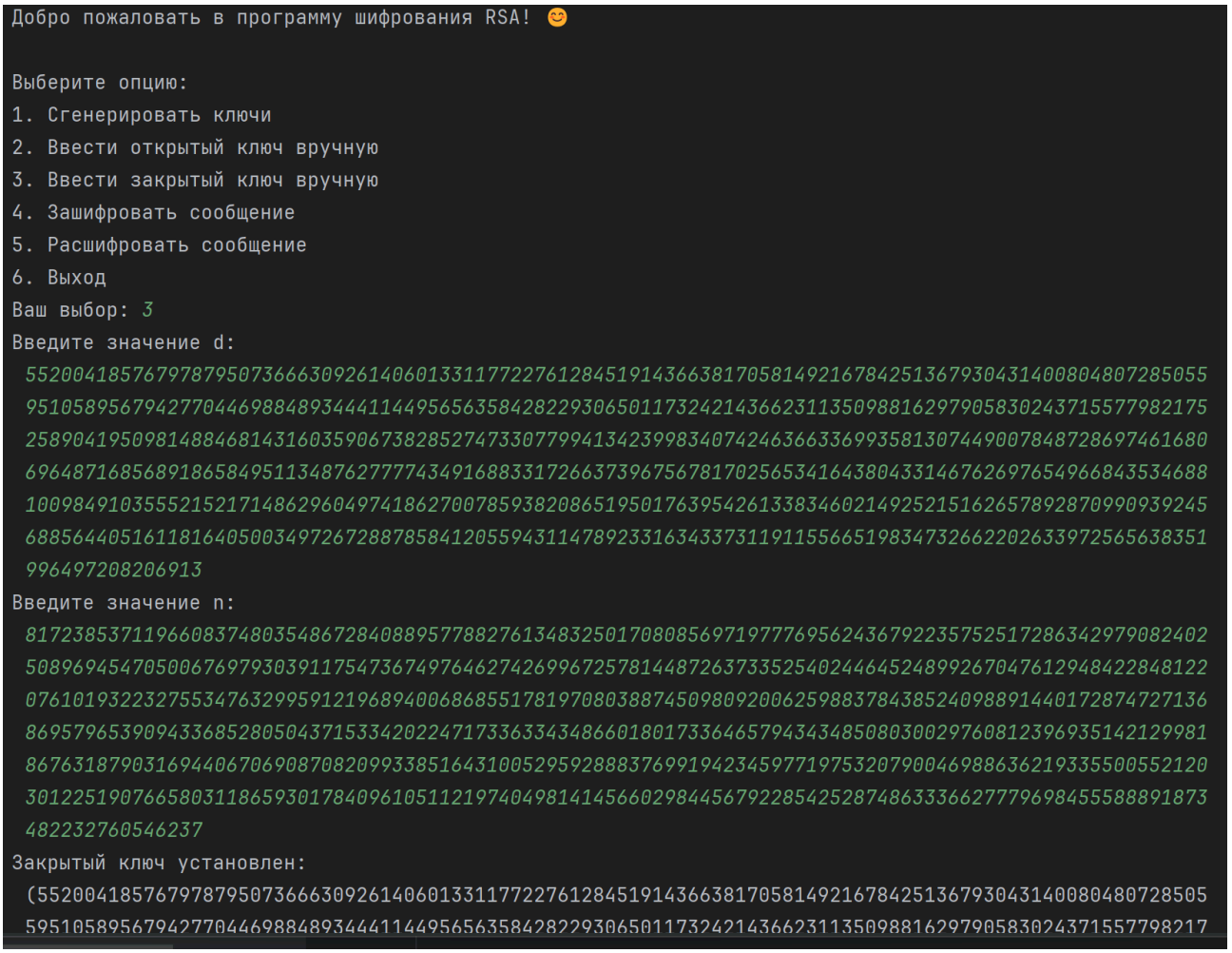

Рисунок 2.4 - Установка закрытого ключа (1024 бит)

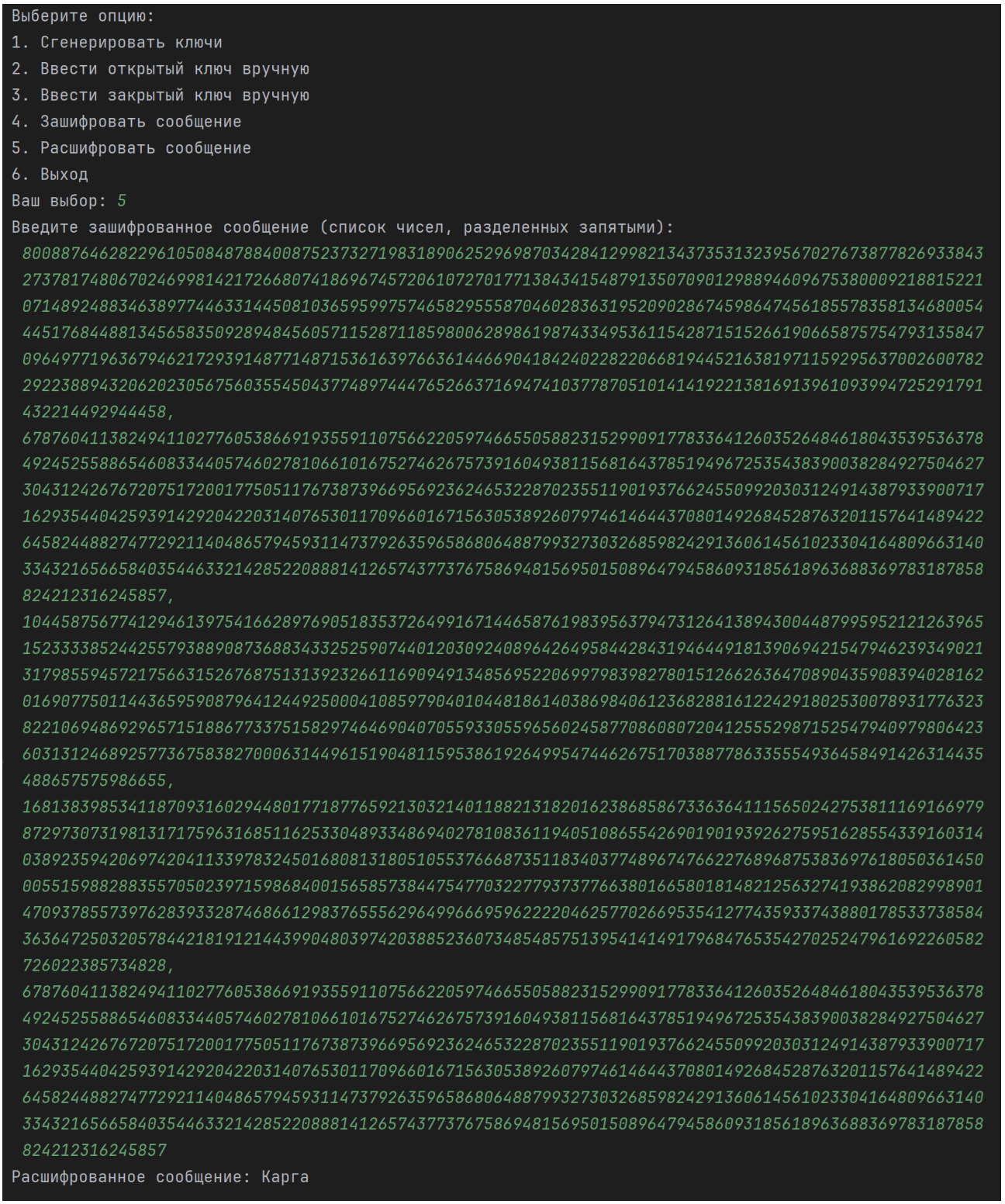

Рисунок 2.5 - Расшифровка сообщения (1024 бит)

Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: понятие шифрование; ассимметричный алгоритм шифрования RSA; дружелюбный консольный интерфейс. Написанная программа была протестирована, полученный результат соответствует значению в примере.