

Лабораторная работа № 2.

Изучение алгоритма RSA

Карасев Илья Алексеевич, М23-505

Цель работы, постановка задачи

Освоить механизм шифрования и дешифрования данных в криптографической системе с открытыми ключами RSA.

Задача:

1. Разработать программу, осуществляющую шифрование и дешифрование сообщения алгоритмом RSA. Ключи генерируются на основе чисел p и q , значения которых выбирается из таблицы 4.1 в соответствии с вариантом. При выборе числа e использовать минимально возможное
2. Исходное сообщение M может состоять из символов. как русского, так и любого другого алфавита.
3. Обеспечить вывод ключей и зашифрованного текста.
4. В программе предусмотреть проверку, являются ли два числа взаимно простыми.

Описание исходных данных

Алгоритм ассиметричного шифрования RSA с заданными значениями p и q

Вариант №12

$p = 167$

$q = 401$

Используемый язык Python

Алгоритм работы программы

1. На вход программе дается строка для шифрования и дешифрования, опционально можно указать иные параметры p и q , если они не даны то используются значения из варианта задания
2. Если значения p и q отличаются от значений по умолчанию, то делается проверка, являются заданные значения простыми целыми числами. Если проверка не проходит выводится ошибка
3. Генерируются закрытый и открытый ключ (метод `gen_keys()`) и выводятся в консоль
4. Используя пару открытого ключа $\{n, e\}$ зашифровывается заданная строка, полученный шифротекст выводится в консоль
5. Используя закрытый ключ $\{p, q, d\}$ расшифровывается полученный шифротекст и выводится на экран

```
MINGW64:/e/Cources/МИФИ  ×  +  ▾  -  □  ×

IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/2_rsa
$ python ./cypher_rsa.py -h
usage: Шифрование и дешифрование используя RSA [-h] [-p P] [-q Q] text

positional arguments:
  text          Входной текст

options:
  -h, --help    show this help message and exit
  -p P          Параметр p для RSA
  -q Q          Параметр q для RSA

IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/2_rsa
$ █
```

Текст программы

```
import argparse

class CypherRSA:
    def __init__(self, p=167, q=401) -> None:
        if p == 167 or self.is_prime(p):
            self._p = p
        else:
            raise ValueError(
                f"ОШИБКА: Параметр p должен быть простым числом. Заданное p={p}"
            )

        if q == 401 or self.is_prime(q):
            self._q = q
        else:
            raise ValueError(
                f"ОШИБКА: Параметр q должен быть простым числом. Заданное q={q}"
            )

        self.e = 0
        self.n = 0
        self._d = None

        self.gen_keys()

    @classmethod
    def is_prime(cls, a):
        """Проверка является ли число простым"""

        if a < 2:
            return False

        for i in range(2, int(a**0.5) + 1):
            if a % i == 0:
                return False

        return True

    @classmethod
    def is_coprime(cls, a, b):
        """Проверка, являются ли два числа взаимно простыми"""

        while b:
            a, b = b, a % b

        return a == 1

    def gen_keys(self):
        """Генерация ключей"""
```

```

n = self._p * self._q
phi = (self._p - 1) * (self._q - 1)

e = 2

while e < phi:
    if self.is_coprime(e, phi):
        break
    e += 1

self.n = n
self.e = e

d = 0

while True:
    if (d * e) % phi == 1:
        break
    d += 1

self._d = d

def encrypt(self, text: str):
    """Зашифровывает заданный текст"""
    encrypted = [pow(ord(c), self.e, self.n) for c in text]
    return encrypted

def decrypt(self, message) -> str:
    """Расшифровывает заданный шифротекст"""
    decrypted = [chr(pow(c, self._d, self.n)) for c in message]
    return "".join(decrypted)

def print_pub_key(self):
    print(f"Публичный ключ: {{ n: {self.n}, e: {self.e} }}")

def print_private_key(self):
    print(f"Закрытый ключ: {{ p:{self._p}, q:{self._q}, d:{self._d} }}")

parser = argparse.ArgumentParser(
    prog="Шифрование и дешифрование используя RSA",
)

parser.add_argument(
    "text",
    type=str,
    help="Входной текст",
)

parser.add_argument(
    "-p",
    type=int,
    action="store",
    help="Параметр p для RSA",
    default=167,
)

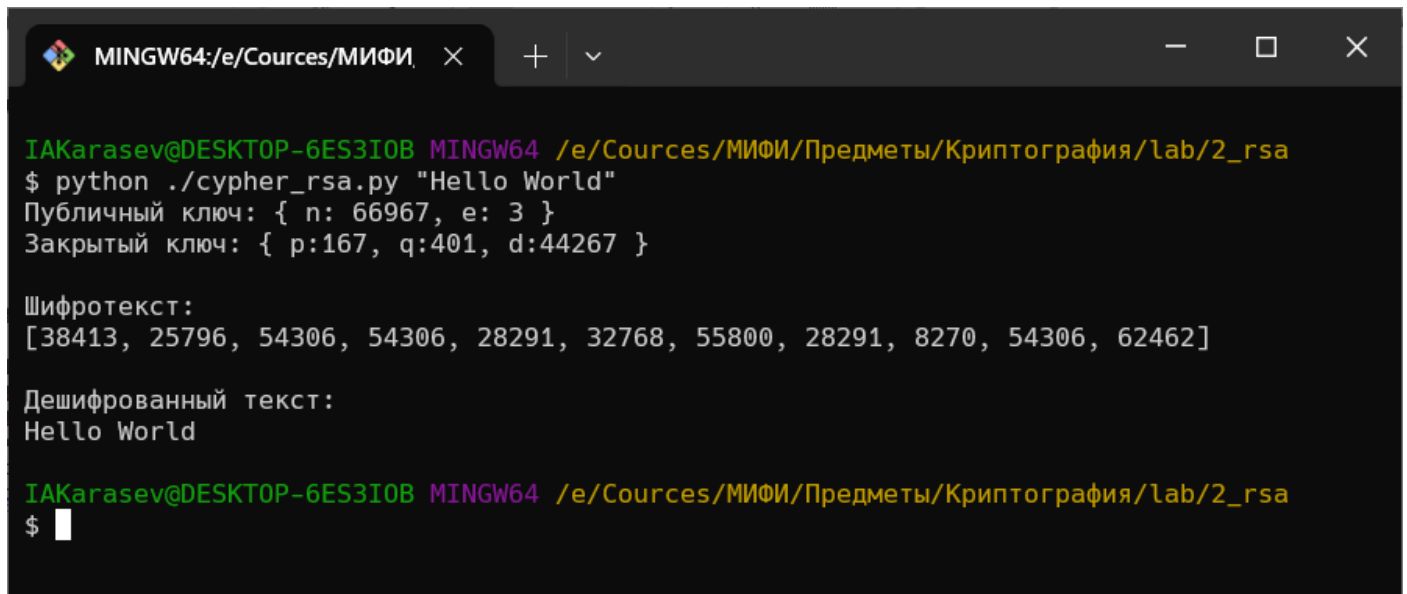
parser.add_argument(
    "-q",
    type=int,
    action="store",
    help="Параметр q для RSA",
    default=401,
)

args = parser.parse_args()

cypher = CypherRSA(p=args.p, q=args.q)
cypher.print_pub_key()
cypher.print_private_key()
enc = cypher.encrypt(args.text)
print(f"\nШифротекст: \n{enc}\n")
print(f"Дешифрованный текст: \n{cypher.decrypt(enc)}")

```

Результаты работы программы



```
IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/2_rsa
$ python ./cypher_rsa.py "Hello World"
Публичный ключ: { n: 66967, e: 3 }
Закрытый ключ: { p:167, q:401, d:44267 }

Шифротекст:
[38413, 25796, 54306, 54306, 28291, 32768, 55800, 28291, 8270, 54306, 62462]

Дешифрованный текст:
Hello World

IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/2_rsa
$
```

Анализ результатов

В результате шифрования каждый символ зашифровывается отдельно, итоговый шифротекст представляет собой массив чисел.

При дешифровании шифротекста с использованием соответствующего закрытого ключа получаем исходную строку

Выводы

Более подробно изучили алгоритм работы RSA. Реализовали данный алгоритм в виде программы на Python.

Контрольные вопросы

1. Что такое однонаправленные функции?

Это функция, обладающая следующим свойством: если известен аргумент функции, то значение функции легко вычисляется, но из значения функции сложно вычислить исходный аргумент

2. Основные свойства однонаправленных функций с потайным ходом.

Однонаправленность: зная x легко вычисляется $f(x)$, при этом при известном значении $f(x)$ сложно или невозможно вычислить x

Потайной ход – дополнительная информация, которая позволяет эффективно вычислять x при наличии значения $f(x)$

Алгоритмы основанные на однонаправленных функциях в основном являются стойкими к атакам перебором, анализом времени вычисления и ресурсов при неизвестном потайном ходе. Данные являющиеся потайным ходом должны быть защищены от распространения так же, как и закрытый ключ.

3. Какие числа называются взаимно простыми?

Два числа называются взаимно простыми, если наибольший общий делитель обоих чисел равен 1

5. Как реализуется программное возведение в степень для больших чисел?

Одним из примеров является метод быстрого возведения в степень. Этот метод основан на рекурсивном принципе и позволяет уменьшить количество операций. Он реализуется

следующим образом:

Если степень n четная, то $a^n = (a^{n/2})^2$.

Если степень n нечетная, то $a^n = a * a^{(n-1)}$.

Так же широко распространены метод модульного возведение в степень и алгоритмы основанные на быстром преобразовании Фурье (FFT)

6. На чем основана криптостойкость алгоритма RSA?

Основной фактор стойкости RSA – сложность операции факторизации больших целых чисел.

Разложение большого числа на простые множители является вычислительно тяжелой задачей.

7. Каковы достоинства и недостатки асимметричных алгоритмов?

Достоинства: Безопасность передачи открытого ключа (отсутствие необходимости секретного обмена ключами); Возможность использования данных алгоритмов в цифровых подписях и аутентификации; Возможность использования данных алгоритмов для обмена ключами в протоколах безопасности

Недостатки: Обычно более вычислительно сложны, чем симметричные, из чего следует ограниченная производительность ассиметричных алгоритмов; Плохая производительность на больших объёмах данных; Необходимость использования длинных ключей для обеспечения стойкости к атакам методом подбора, факторизации и других; Более сложная и трудоемкая реализация по сравнению с симметричными алгоритмами.