Implementacja publicznego systemu kryptograficznego w oparciu o algorytm RSA

Wykonał: Dominik Łukasiewicz, 145290

1. **Opis RSA:**
   RSA jest to asymetryczny algorytm kryptograficzny służący do szyfrowania danych. Wykorzystuje on klucz publiczny.
2. **Założenia:**
   Program został napisany w języku python w którym nie ma ograniczenia co do wielkości wartości int, a więc teoretycznie dopóki mamy miejsce w pamięci jesteśmy w stanie zwiększać obliczane liczby. Warto lecz pamiętać, że dla nawet niewielkich wartości liczb pierwszych (przykładowo w wielkości 10000) program będzie wykonywał obliczenia długo.
3. **Opis metody użytych do wyznaczenia e i d:**
   Do wyznaczenia **e** sprawdzamy zakres od 2 do wartości ϕ(n). Jeśli wybrana wartość oraz ϕ(n) mają największy wspólny dzielnik wynosi 1 to wybieramy tą wartość. Fragment kodu w którym wykonywana jest opisana wyżej metoda:

```
e = 2
for k in range(2,totient):
    if gcd(k,totient)==1:
        e = k
        break
```

Gdzie gcd:

```
def gcd(m,n):
    if n==0:
        return m
    else:
        return gcd(n,m%n)
```

Do wyznaczenia **d** wykorzystujemy wcześniej obliczone **e**. Z zakresu od 1 do momentu uzyskania odpowiedzi wyznaczamy wartość 1 + iteracja * ϕ(n). Jeśli wyznaczona wartość daje resztę 0 po wykonaniu modulo z **e** to przerywany obliczenia i do **d** przepisujemy wartość 1 + iteracja * ϕ(n) / **e**, z czego musimy zapewnić, że wynikiem tego będzie liczba całkowitoliczbowa.

Fragment kodu w którym wykonywana jest opisana wyżej metoda:

```
d = 0
i = 1
while(True):
    x = 1 + i*totient
    i += 1
    if x % e == 0:
        d = int(x/e)
        break
```

4. Opis realizacji zadań:

- Znaki wejściowe (plik tests\1\original.txt)

```
1    jakas wiadomosc do zaszyfrowania oraz do odszyfrowania
```

- Wartości p, q, e, d oraz n

```
p: 1759, q: 1663
e: 5, d: 2337437, n: 2925217
```

- Znaki zaszyfrowane (fragment) (plik tests\1\encrypted.txt)

```
1    2313218 1828362 2027009 1828362 2705000 1377045 2541530 93854 1828362 1608294 1331631 2523346 1331631 2705000 20032 1377045 1608294
```

- Znaki odszyfrowane (plik tests\1\decrypted.txt)

```
1    jakas wiadomosc do zaszyfrowania oraz do odszyfrowania
```

5. Odpowiedz na pytania:

- Jakie elementy algorytmu są trudne w realizacji? – Najbardziej kosztownym fragmentem algorytmu jest odszyfrowywanie wiadomości. Wynika to z potrzeby spotęgowania wartości zaszyfrowanej do potęgi **d**, gdzie **d** może osiągać duże wartości. Poza tym na uzyskanym wyniku trzeba przeprowadzić modulo z **n**.
- Co stanowi o bezpieczeństwie i jakości tego algorytmu szyfrowania? – Bezpieczeństwo RSA leży w trudności faktoryzacji dużych wartości. Mnożenie dwóch ogromnych liczb nie stanowi problemu, lecz znalezienie dzielnika już tak.

6. Wnioski:

- Dla nawet niewielkich wartości liczb pierwszych (przykładowo około 10000) wykonanie odszyfrowania przez program zajmuje ogromną ilość czas.
- Samo szyfrowanie wiadomości nie stanowi większego problemu jeśli chodzi o potrzebny czas do zaszyfrowania.