Zadanie RSA - wersja podstawowa

Proszę uzupełnić:

Nazwisko: Zmiendak

Imię: Wiktor

Nr Albumu: 142706

Proszę Uzupełnić powyżej dane osobowe!

Uwaga:

Na dielcie należy umieścić notatnik zapisany w trzech formatach `.ipynb` oraz `.html` oraz `.pdf` , pliki muszą mieć nazwy:

`Nazwisko_Imie_nrAlbumu.*`

Temat

Niech $x_0 \ldots x_7$ będzie dowolną permutacją ciągu cyfr daty urodzenia oaoby rozwiązującej zadanie, w której $x_0 \neq 0$ i $x_4 \neq 0$.

p jest trzecią liczbą pierwszą następującą po liczbie $x_0x_1x_2x_3$ a q jest piątą liczbą pierwszą następującą po liczbie $x_4x_5x_6x_7$

Uwaga: Jeżeli $p = q$ to zmienić permutację cyfr x_i lub wybrać kolejne liczby pierwsze np. 4 i 6 tak aby $p \neq q$

$$n = p \cdot q$$

- Wyznacz wszystkie pary liczb, które mogą być kluczami dla RSA
- Wyznacz liczbę par kluczy takich że $e = d \mod \Phi(n)$
- Wybierz dowolną parę kluczy e i d takich, że $e \neq d \mod \Phi(n)$
- Niech $m = \texttt{randint(10^3, min(10^10,n))}$, wykorzystując wybrane w poprzednim punkcie klucze zaszyfruj m, sprawdź czy deszyfrowanie daje poprawny wynik i podpisz cyfrowo m (wykorzystując algorytm podpisu RSA)
- Sprawdź że jest w pierścieniu $\mathbb{Z}_{\Phi(n)}$ elementów odwracalnych e, takich, że $e = d \mod \Phi(n)$
- Zbadaj czas potrzebny na faktoryzację n przy założeniu, że znana jest wartość $\Phi(n)$ **Uwaga: należy wyznaczać rozkład n bez wykorzystywania funkcji `factor` itp.,. ! W tym przypadku rząd wielości n jest na tyle mały, że funkcja `factor` prawdopodobnie wyznaczy rozkład n Należy podać metodę faktoryzacji wykorzystującą wartość $\Phi(n)$**
- Notatnik należy przygotować tak aby były zachowane dane losowe, dla których zostały wykonane obliczenia i była możliwość ponownego przeliczenia rozwiązań dla tych wartości oraz aby było możliwe wygenerowanie nowych liczb spełniających warunki z tematu i wyznaczenie rozwiązań. Całość powinna zwracać poprawne wyniki po wybraniu opcji jupytera Cell Run All

Rozwiązanie

In [86]:
```
import random
import time

#Numbers from the date of my birthday
num_list = [1,5,1,1,2,0,0,2]

birth_list = ''.join(str(item) for item in num_list)

first_num = Integer(birth_list[:4])
second_num = Integer(birth_list[4:])

print(f"First value: {first_num}\nSecond value: {second_num}")

#Values preparation
p = first_num
q = second_num

for i in range(3):
    p = next_prime(p)

for i in range(5):
    q = next_prime(q)

n = p * q

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

print(f"P: {p}\nQ: {q}\nN: {n}\nPhi: {phi}")

#Keys finding
keys = []

for i in range(2, phi):
    if gcd(i, phi) == 1:
        e = i
        d = Integer(i).inverse_mod(phi)
        keys.append((e,d))

for pair in keys:
    e, d = pair
    if e == (d%phi):
        print(f"E = {e}, D = {d}")

#Sorting keys
valid_keys = [(e,d) for e,d in keys if e != (d%phi)]

#Random pair of keys choice
random_key = random.choice(valid_keys)
e, d = random_key

print(f"Random generated key: E = {e}, D = {d}")

#Factoryzation
def factorization(n):
    a = ceil(sqrt(n))
    b2 = a^2 - n
    while not is_square(b2):
        a += 1
        b2 = a^2 - n
    p = a + sqrt(b2)
    q = a - sqrt(b2)

    return int(p), int(q)

#Couting time of factorization
start_time = time.time()
p_factor, q_factor = factorization(n)
end_time = time.time()

factorization_time = end_time - start_time

print(f"Time of factorization: {factorization_time}")

#Random value
m = randint(10^3, min(10^10, n))
```

```
First value: 1511
Second value: 2002
P: 1543
Q: 2029
N: 3130747
Phi: 3127176
E = 86867, D = 86867
E = 149059, D = 149059
E = 235925, D = 235925
E = 545869, D = 545869
E = 632735, D = 632735
E = 694927, D = 694927
E = 781793, D = 781793
E = 781795, D = 781795
E = 868661, D = 868661
E = 930853, D = 930853
E = 1017719, D = 1017719
E = 1327663, D = 1327663
E = 1414529, D = 1414529
E = 1476721, D = 1476721
E = 1563587, D = 1563587
E = 1563589, D = 1563589
E = 1650455, D = 1650455
E = 1712647, D = 1712647
E = 1799513, D = 1799513
E = 2109457, D = 2109457
E = 2196323, D = 2196323
E = 2258515, D = 2258515
E = 2345381, D = 2345381
E = 2345383, D = 2345383
E = 2432249, D = 2432249
E = 2494441, D = 2494441
E = 2581307, D = 2581307
E = 2891251, D = 2891251
E = 2978117, D = 2978117
E = 3040309, D = 3040309
E = 3127175, D = 3127175
Random generated key: E = 3001433, D = 1142633
Time of factorization: 0.0002138614654541016
```

Obliczenia dla wartości losowych wygenerowanych podczas przygotowania rozwiązań

(Należy zapisać otrzymane dane)

In [87]:
```
#Crypting and encrypting
print(f"M: {m}")

c = pow(m, e, n)

m = pow(c, d, n)

print(f"Crypted message: {c}\nEncrypted message: {m}")
```

```
M: 1074938
Crypted message: 1286842
Encrypted message: 1074938
```

Program pozwalający wygenerować nowe dane spełniające założenia z tematu i wyznaczenie odpowiedzi

In [88]:
```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

num_list = []

#Values preperation
for i in range(8):
    if i == 3 or i == 0:
        num_list.append(random.choice(numbers[1:]))
    else:
        num_list.append(random.choice(numbers))

num_list = ''.join(str(item) for item in num_list)

p = Integer(num_list[:4])

q = Integer(num_list[4:])

print(f"Starting values: {p}, {q}")

for i in range(3):
    p = next_prime(p)

for i in range(5):
    q = next_prime(q)

n = p * q

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

print(f"P: {p}\nQ: {q}\nN: {n}\nPhi: {phi}")

keys = []

for i in range(2, phi):
    if gcd(i, phi) == 1:
        e = i
        d = Integer(i).inverse_mod(phi)
        keys.append((e,d))

for pair in keys:
    e, d = pair
    if e == (d%phi):
        print(f"E = {e}, D = {d}")

valid_keys = [(e,d) for e,d in keys if e != (d%phi)]

random_key = random.choice(valid_keys)
e, d = random_key

print(f"Random generated key: {random_key}")

m = randint(10^3, min(10^10, n))

print(f"M: {m}")

c = pow(m, e, n)

m = pow(c, d, n)

print(f"Crypted message: {c}\nEncrypted message: {m}")
```

```
Starting values: 7701, 2324
P: 7723
Q: 2351
N: 18156773
Phi: 18146700
E = 439451, D = 439451
E = 649351, D = 649351
E = 683099, D = 683099
E = 1088801, D = 1088801
E = 1702999, D = 1702999
E = 1771901, D = 1771901
E = 2157299, D = 2157299
E = 2352349, D = 2352349
E = 2791801, D = 2791801
E = 3035449, D = 3035449
E = 3246101, D = 3246101
E = 3441151, D = 3441151
E = 3474899, D = 3474899
E = 4124251, D = 4124251
E = 4509649, D = 4509649
E = 4563701, D = 4563701
E = 4949099, D = 4949099
E = 5598451, D = 5598451
E = 5632199, D = 5632199
E = 5827249, D = 5827249
E = 6037901, D = 6037901
E = 6281549, D = 6281549
E = 6721001, D = 6721001
E = 6916051, D = 6916051
E = 7301449, D = 7301449
E = 7370351, D = 7370351
E = 7984549, D = 7984549
E = 8390251, D = 8390251
E = 8423999, D = 8423999
E = 8633899, D = 8633899
E = 9073349, D = 9073349
E = 9073351, D = 9073351
E = 9512801, D = 9512801
E = 9722701, D = 9722701
E = 9756449, D = 9756449
E = 10162151, D = 10162151
E = 10776349, D = 10776349
E = 10845251, D = 10845251
E = 11238649, D = 11238649
E = 11425699, D = 11425699
E = 11865151, D = 11865151
E = 12108799, D = 12108799
E = 12319451, D = 12319451
E = 12514501, D = 12514501
E = 12548249, D = 12548249
E = 13197601, D = 13197601
E = 13582999, D = 13582999
E = 13637051, D = 13637051
E = 14022449, D = 14022449
E = 14071801, D = 14071801
E = 14705549, D = 14705549
E = 14900599, D = 14900599
E = 15111251, D = 15111251
E = 15354899, D = 15354899
E = 15794351, D = 15794351
E = 15989401, D = 15989401
E = 16374799, D = 16374799
E = 16443701, D = 16443701
E = 17057899, D = 17057899
E = 17463601, D = 17463601
E = 17497349, D = 17497349
E = 17707249, D = 17707249
E = 18146699, D = 18146699
Random generated key: (14725283, 17177447)
M: 3537517
Crypted message: 13113759
Encrypted message: 3537517
```

In [ ]:
```

```