In [1]:

```
import random
import sys
import hashlib
from typing import Callable, Tuple

sys.setrecursionlimit(100000)

from random import randrange

def gcd(a: int, b: int) -> int:
    """Calculates greatest common divisor of 2 numbers"""

    if (a == 0):
        return b
    return gcd(b % a, a)

def generate_relative_coprime(prime_to: int, get_candidate: Callable[[], int]) ->
 int:
    """
    Generates a relatevely prime number
    """

    commond_divisor = 0
    candidate = 0

    while commond_divisor != 1:
        candidate = get_candidate()
        commond_divisor = gcd(candidate, prime_to)

    return candidate

def get_relative_by_module(d, z):
    candidate = 0
    while (candidate * d) % z != 1:
        candidate = random.randrange(3, z, 2)

    return candidate

def generate_keypair(p: int, q: int) -> tuple:
    """
    Generates private and public keys using RSA algorithm.

    :param p: A prime number
    :param q: A prime number
    """

    n = p * q
    z = (p - 1) * (q - 1)

    d = generate_relative_coprime(z, lambda: random.randrange(3, z, 2))
    e = get_relative_by_module(d, z)

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

def encrypt(public_key: Tuple[int, int], plaintext: str) -> list:
    """
    Encrypts the message using public key
    """

    e, n = public_key
    cipher = [(ord(char) ** e) % n for char in plaintext]

    return cipher

def decrypt(private_key: Tuple[int, int], ciphertext: list) -> str:
    """
    Decrypts message using private key
    """
```

```
    d, n = private_key
    plain = [chr((char ** d) % n) for char in ciphertext]

    return ''.join(plain)

public, private = generate_keypair(859, 733)
print("Alise ģenerē publisko un privāto atslēgu pāri.")
print("Alise dalās ar savu publisko atslēgu: ", public)

message = 'KINO'
message_hash = hashlib.md5(message.encode('utf-8')).hexdigest()
print("Alise veido vēstules MD5 hešu un šifrē to ar privāto atslēgu.")
print("Alise nosūta 2 vēstules Bobam.")
print("Atklāta vēstule:", message)

cipher = encrypt(private, message_hash)
print("Vēstules heša šifrs: ", ''.join(str(x) for x in cipher))

print("Bobs saņem 2 vēstules.")
print("Bobs atšifrē vēstuli ar publisko atslēgu: ", decrypt(public, cipher))
print("Bobs izveido atklāta vēstules MD5 hešu: ", message_hash)
print("Bobs salīdzina vēstules hešu ar atšifrēto vēstuli. Ja tie sakrīt, sūtītājs
 un ziņojums ir pareizi.")
```

```
Alise ģenerē publisko un privāto atslēgu pāri.
Alise dalās ar savu publisko atslēgu:  (324809, 629647)
Alise veido vēstules MD5 hešu un šifrē to ar privāto atslēgu.
Alise nosūta 2 vēstules Bobam.
Atklāta vēstule: KINO
Vēstules heša šifrs:  37427445606145606137427421436922914426619316190445606157116526619318794937427437427453599216190424203214369266193312595187949410137571165410137434895571165242032420343489531259537427457116 5
Bobs saņem 2 vēstules.
Bobs atšifrē vēstuli ar publisko atslēgu:  5225dc492f4755a91d4e76f60f110e5f
Bobs izveido atklāta vēstules MD5 hešu:  5225dc492f4755a91d4e76f60f110e5f
Bobs salīdzina vēstules hešu ar atšifrēto vēstuli. Ja tie sakrīt, sūtītājs un zi
ņojums ir pareizi.
```