

INSTITUTE OF COMPUTER TECHNOLOGY
B-TECH COMPUTER SCIENCE ENGINEERING 2025-26
SUBJECT:-CRYPTOGRAPHY

NAME: Rahul Prajapati

ENRLL NO: 23162171020

BRANCH: CYBER SECURITY

BATCH: 52

PRACTICAL_9

Aim: To understand the fundamentals of Public Key Cryptography by implementing the RSA algorithm for encryption and decryption, where a public key is used to encrypt a user-inputted message and a private key is used to decrypt it back to its original form, displaying both ciphertext and decrypted text in the console.

CODE:

```
 praktical9_1.py > is_prime
 1  def gcd(a, b):
 2      while b:
 3          a, b = b, a % b
 4      return a
 5
 6  def is_prime(n):
 7      if n <= 1:
 8          return False
 9      if n <= 3:
10          return True
11      if n % 2 == 0 or n % 3 == 0:
12          return False
13      i = 5
14      while i * i <= n:
15          if n % i == 0 or n % (i + 2) == 0:
16              return False
17          i += 6
18      return True
19
20  def modinv(a, m):
21      m0 = m
22      x0, x1 = 0, 1
23      while a > 1:
24          q = a // m
25          a, m = m, a % m
26          x0, x1 = x1 - q * x0, x0
27      if x1 < 0:
28          x1 += m0
29      return x1
30
31  def generate_keypair(p, q):
32      n = p * q
33      phi = (p - 1) * (q - 1)
34      e = 3
35      while gcd(e, phi) != 1:
36          e += 2
37      d = modinv(e, phi)
38      return ((e, n), (d, n))
39
40  def encrypt(public_key, plaintext):
41      e, n = public_key
42      cipher = [pow(ord(char), e, n) for char in plaintext]
43      return cipher
44
45  def decrypt(private_key, ciphertext):
46      d, n = private_key
47      plain = [chr(pow(char, d, n)) for char in ciphertext]
48      return ''.join(plain)
49
```

```

49
50     if __name__ == "__main__":
51         try:
52             p = int(input("Enter a prime number (p): "))
53             q = int(input("Enter another prime number (q): "))
54         except ValueError:
55             print("Invalid input. Please enter integers.")
56             exit(1)
57
58         if not (is_prime(p) and is_prime(q)):
59             print("Both numbers must be prime.")
60             exit(1)
61
62         n = p * q
63         plaintext = input("Enter plaintext: ")
64
65         for char in plaintext:
66             if ord(char) >= n:
67                 print(f"Character '{char}' (ord={ord(char)}) is too large for modulus n={n}")
68                 print("Use larger prime numbers so n > 127.")
69                 exit(1)
70
71         public_key, private_key = generate_keypair(p, q)
72         print("Public Key:", public_key)
73         print("Private Key:", private_key)
74
75         ciphertext = encrypt(public_key, plaintext)
76         print("Ciphertext:", ' '.join(map(str, ciphertext)))
77
78         recovered_text = decrypt(private_key, ciphertext)
79         print("Recovered Plaintext:", recovered_text)
80
81         wrong_private_key = (private_key[0] + 1, private_key[1])
82         try:
83             wrong_recovered_text = decrypt(wrong_private_key, ciphertext)
84             print("Wrongly Recovered Plaintext:", wrong_recovered_text)
85         except Exception as e:
86             print("Decryption failed with wrong key:", str(e))
87         correct_recovered_text = decrypt(private_key, ciphertext)
88         print("Correctly Recovered Plaintext:", correct_recovered_text)
89

```

OUTPUT:

TERMINAL

```

PS C:\Users\Hp\python .\practical9_1.pyCryptography\Practicals_source_code>
Enter a prime number (p): 97
Enter another prime number (q): 101
Enter plaintext: hello_world
Public Key: (7, 9797)
Private Key: (2743, 9797)
Ciphertext: 1177 2222 1908 1908 7969 5692 3179 7969 7721 1908 6261
Recovered Plaintext: hello_world
Wrongly Recovered Plaintext: ¶ÑÓææ[Ñ·
Correctly Recovered Plaintext: hello_world
PS C:\Users\Hp\OneDrive\Desktop\SEM_05\Cryptography\Practicals_source_code>

```