**Q.No.17 Write a high-level code for RSA public-key encryption scheme, each user has a public key, e, and a private key, d. Suppose Bob leaks his private key. Rather than generating a new modulus, he decides to generate a new public and a new private key. Is this safe?**

**CODE :**

**import random**

**def is\_prime(num):**

**if num == 2:**

**return True**

**if num < 2 or num % 2 == 0:**

**return False**

**for n in range(3, int(num \*\* 0.5) + 2, 2):**

**if num % n == 0:**

**return False**

**return True**

**def gcd(a, b):**

**while b != 0:**

**a, b = b, a % b**

**return a**

**def generate\_keypair(p, q):**

**if not (is\_prime(p) and is\_prime(q)):**

**raise ValueError("Both numbers must be prime.")**

**elif p == q:**

**raise ValueError("p and q cannot be equal.")**

**n = p \* q**

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

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**while g != 1:**

**e = random.randrange(1, phi)**

**g = gcd(e, phi)**

**d = multiplicative\_inverse(e, phi)**

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

**def multiplicative\_inverse(e, phi):**

**d = 0**

**x1 = 0**

**x2 = 1**

**y1 = 1**

**temp\_phi = phi**

**while e > 0:**

**temp1 = temp\_phi // e**

**temp2 = temp\_phi - temp1 \* e**

**temp\_phi = e**

**e = temp2**

**x = x2 - temp1 \* x1**

**y = d - temp1 \* y1**

**x2 = x1**

**x1 = x**

**d = y1**

**y1 = y**

**if temp\_phi == 1:**

**return d + phi**

**def encrypt(pk, plaintext):**

**key, n = pk**

**cipher = [(ord(char) \*\* key) % n for char in plaintext]**

**return cipher**

**def decrypt(pk, ciphertext):**

**key, n = pk**

**plain = [chr((char \*\* key) % n) for char in ciphertext]**

**return ''.join(plain)**

**# Example usage**

**p = 61**

**q = 53**

**public, private = generate\_keypair(p, q)**

**print("Public key:", public)**

**print("Private key:", private)**

**message = "Hello World"**

**encrypted\_message = encrypt(public, message)**

**print("Encrypted message:", ''.join(map(str, encrypted\_message)))**

**decrypted\_message = decrypt(private, encrypted\_message)**

**print("Decrypted message:", decrypted\_message)**

**OUPUT :**

