Network Security and Forensics

Lab Session 7

Submitted To:-

Dr. Lokesh Chauhan Sir

Submitted By:-

Saloni Rangari

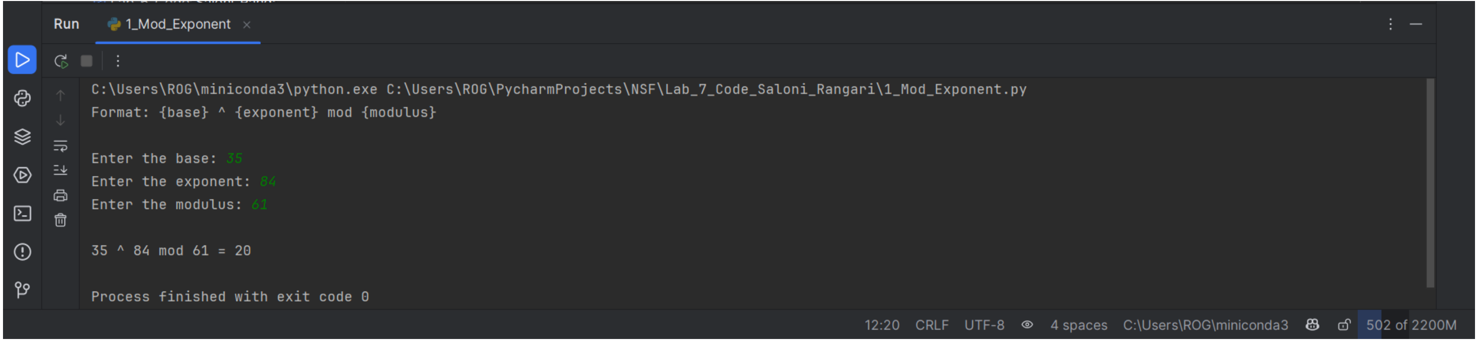
M.Tech. AIDS

Assignment 1: Write a program to calculate the mod exponent of a big number.

```
def mod_exponent(base, exp, mod):
    result = 1
    base = base % mod
    while exp > 0:
        if (exp % 2) == 1:  # If exp is odd
            result = (result * base) % mod
        exp = exp >> 1  # Divide exp by 2
        base = (base * base) % mod
    return result

if __name__ == "__main__":
    base = int(input("Enter the base: "))
    exp = int(input("Enter the exponent: "))
    mod = int(input("Enter the modulus: "))
    print(f"{base}^{exp} mod {mod} = {mod_exponent(base, exp, mod)}")
```

Output:


Run
1_Mod_Exponent
C:\Users\ROG\miniconda3\python.exe C:\Users\ROG\PycharmProjects\NSF\Lab_7_Code_Saloni_Rangari\1_Mod_Exponent.py
Format: {base} ^ {exponent} mod {modulus}
Enter the base: 35
Enter the exponent: 84
Enter the modulus: 61
35 ^ 84 mod 61 = 20
Process finished with exit code 0
12:20
CRLF
UTF-8
4 spaces
C:\Users\ROG\miniconda3
502 of 2200M

Assignment 2: Write a program to demonstrate RSA algorithm by taking input plaintext from a file and produce output file into ciphertext file.

```
import random
from sympy import isprime

def generate_prime_candidate(length):
    p = random.getrandbits(length)
    return p | (1 << length - 1) | 1  # Ensure p is odd and has the correct
length

def generate_prime_number(length):
    p = 4
    while not isprime(p):
        p = generate_prime_candidate(length)
    return p

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

def multiplicative_inverse(e, phi):
    d_old, d_new = 0, 1
    r_old, r_new = phi, e
    while r_new > 0:
        quotient = r_old // r_new
        d_old, d_new = d_new, d_old - quotient * d_new
        r_old, r_new = r_new, r_old - quotient * r_new
    if r_old > 1:
        raise Exception('No modular inverse')
    if d_old < 0:
        d_old += phi
    return d_old

def rsa_keypair(length):
    p = generate_prime_number(length)
    q = generate_prime_number(length)

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

    e = random.randrange(2, phi)

    while gcd(e, phi) != 1:
        e = random.randrange(2, phi)

    d = multiplicative_inverse(e, phi)

    return ((e, n), (d, n))  # Public key and private key

def encrypt(plaintext, pubkey):
    e, n = pubkey
    plaintext_int = int.from_bytes(plaintext.encode(), 'big')
    ciphertext_int = pow(plaintext_int, e, n)
    return ciphertext_int

def decrypt(ciphertext_int, privkey):
    d, n = privkey
    plaintext_int = pow(ciphertext_int, d, n)
    plaintext_bytes = plaintext_int.to_bytes((plaintext_int.bit_length() + 7) //
8, 'big')
    return plaintext_bytes.decode()

if __name__ == "__main__":
    length = int(input("Enter the key length (in bits): "))

    public_key, private_key = rsa_keypair(length)

    with open('plaintext.txt', 'r') as file:
        plaintext = file.read().strip()

    ciphertext_int = encrypt(plaintext, public_key)

    with open('ciphertext.txt', 'w') as file:
        file.write(str(ciphertext_int))

    print(f"Ciphertext: {ciphertext_int}")

    decrypted_text = decrypt(ciphertext_int, private_key)

   print(f"Decrypted Text: {decrypted_text}")
```

Output:

Run 2_RSA

```
C:\Users\ROG\miniconda3\python.exe C:\Users\ROG\PycharmProjects\NSF\Lab_7_Code_Saloni_Rangari\2_RSA.py
Plain Text: Saloni Rangari

Enter the key length (in bits): 64

Ciphertext: 18228428941542637485162178485806293818
Decrypted Text: Saloni Rangari

Process finished with exit code 0
```

84:15 CRLF UTF-8 4 spaces C:\Users\ROG\miniconda3 940 of 2200M

Assignment 3: Write a program to implement Fermit Little Theorem.

```
def fermat_little_theorem(a, p):
    if p <= 1 or not isprime(p):
        raise ValueError("p must be a prime number greater than 1.")

    return mod_exponent(a, p - 1, p)

if __name__ == "__main__":
    a = int(input("Enter an integer a: "))
    p = int(input("Enter a prime number p: "))

    result = fermat_little_theorem(a, p)

    print(f"{a}^{p-1} mod {p} ≡ {result} (according to Fermat's Little
Theorem)")
```

Output:

Run 3_Fermit_Little_Theorem

```
C:\Users\ROG\miniconda3\python.exe C:\Users\ROG\PycharmProjects\NSF\Lab_7_Code_Saloni_Rangari\3_Fermit_Little_Theorem.py
Enter an integer a: 5
Enter a prime number p: 3
5^2 mod 3 ≡ 1 (according to Fermat's Little Theorem)

Process finished with exit code 0
```

17:35 CRLF UTF-8 4 spaces