**Assingment 10**

1. Write a Python script to list all the prime numbers from 1 to 100 using Sieve of Eratosthenes.

def sieve\_of\_eratosthenes(limit):

# Create a boolean array "prime[0..limit]" and initialize all entries as true

prime = [True] \* (limit + 1)

p = 2

while p \* p <= limit:

# If prime[p] is not changed, it is a prime

if prime[p]:

# Updating all multiples of p to not prime

for i in range(p \* p, limit + 1, p):

prime[i] = False

p += 1

# Collect all prime numbers

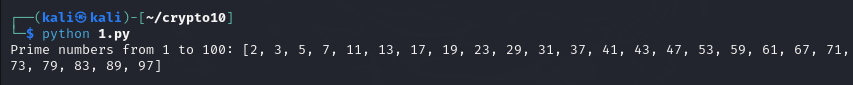
primes = [p for p in range(2, limit + 1) if prime[p]]

return primes

# List all primes from 1 to 100

primes\_up\_to\_100 = sieve\_of\_eratosthenes(100)

print("Prime numbers from 1 to 100:", primes\_up\_to\_100)



1. Write a Python script to implement RSA algorithm using build in functions (both encryption and decryption)

import random

from sympy import mod\_inverse

# RSA key generation

def generate\_keys():

# Select two large primes, for simplicity we'll use smaller primes

p = 61

q = 53

n = p \* q

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

# Choose e such that 1 < e < phi and gcd(e, phi) = 1

e = 17 # Commonly chosen prime

d = mod\_inverse(e, phi)

# Public and private keys

public\_key = (e, n)

private\_key = (d, n)

return public\_key, private\_key

# RSA encryption

def rsa\_encrypt(plaintext, public\_key):

e, n = public\_key

# Convert plaintext to an integer and encrypt

plaintext\_int = int.from\_bytes(plaintext.encode('utf-8'), 'big')

ciphertext = pow(plaintext\_int, e, n)

return ciphertext

# RSA decryption

def rsa\_decrypt(ciphertext, private\_key):

d, n = private\_key

plaintext\_int = pow(ciphertext, d, n)

# Convert decrypted integer back to plaintext, handle decoding error with Latin-1

try:

plaintext = plaintext\_int.to\_bytes((plaintext\_int.bit\_length() + 7) // 8, 'big').decode('utf-8')

except UnicodeDecodeError:

plaintext = plaintext\_int.to\_bytes((plaintext\_int.bit\_length() + 7) // 8, 'big').decode('latin-1')

return plaintext

# Generate keys

public\_key, private\_key = generate\_keys()

print("Public Key:", public\_key)

print("Private Key:", private\_key)

# Encrypt and decrypt a message

message = "HELLO"

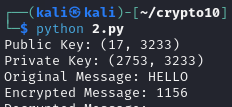
print("Original Message:", message)

ciphertext = rsa\_encrypt(message, public\_key)

print("Encrypted Message:", ciphertext)

decrypted\_message = rsa\_decrypt(ciphertext, private\_key)

print("Decrypted Message:", decrypted\_message)



1. Write a Python script to implement RSA algorithm with out using build in functions (both encryption and decryption)

import random

from sympy import mod\_inverse

# RSA key generation

def generate\_keys():

# Select two large primes, for simplicity we'll use smaller primes

p = 61

q = 53

n = p \* q

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

# Choose e such that 1 < e < phi and gcd(e, phi) = 1

e = 17 # Commonly chosen prime

d = mod\_inverse(e, phi)

# Public and private keys

public\_key = (e, n)

private\_key = (d, n)

return public\_key, private\_key

# RSA encryption

def rsa\_encrypt(plaintext, public\_key):

e, n = public\_key

# Convert plaintext to an integer and encrypt

plaintext\_int = int.from\_bytes(plaintext.encode('utf-8'), 'big')

ciphertext = pow(plaintext\_int, e, n)

return ciphertext

# RSA decryption

def rsa\_decrypt(ciphertext, private\_key):

d, n = private\_key

plaintext\_int = pow(ciphertext, d, n)

# Convert decrypted integer back to plaintext, handle decoding error with Latin-1

try:

plaintext = plaintext\_int.to\_bytes((plaintext\_int.bit\_length() + 7) // 8, 'big').decode('utf-8')

except UnicodeDecodeError:

plaintext = plaintext\_int.to\_bytes((plaintext\_int.bit\_length() + 7) // 8, 'big').decode('latin-1')

return plaintext

# Generate keys

public\_key, private\_key = generate\_keys()

print("Public Key:", public\_key)

print("Private Key:", private\_key)

# Encrypt and decrypt a message

message = "HELLO"

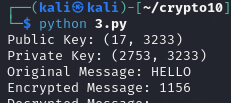
print("Original Message:", message)

ciphertext = rsa\_encrypt(message, public\_key)

print("Encrypted Message:", ciphertext)

decrypted\_message = rsa\_decrypt(ciphertext, private\_key)

print("Decrypted Message:", decrypted\_message)



------------------------------------------------------------------------------------------

“Learning is never done without error, and defeat ” – Vladimir Lenin

“There is no substitute for hard work.” – Thomas Alva Edison

“You cannot change your future, but you can change your habits, and surely your habits will change your future.”– A.P.J. Abdul Kalam

---------------------------------------------------------------------------------------------