CS1702: Network Security

Assignment -1: RSA Implementation

Submitted by: Siddharth Koppisetty

Roll no: CS22B1022

Introduction:

RSA (Rivest–Shamir–Adleman) is a public-key cryptographic algorithm that forms the backbone of many secure communication protocols. It relies on the mathematical hardness of factoring the product of two large prime numbers, a problem for which no efficient solution is known. RSA uses a key pair: a **public key** for encryption and a **private key** for decryption. The keys are generated using modular arithmetic and Euler’s totient function. RSA is widely used in secure data transmission, digital signatures, and key exchange mechanisms. Its asymmetric nature allows it to provide both confidentiality and authentication, making it a critical component of modern network security infrastructures.

Code (rsa-implementation.py):

import random

import sympy

def generate\_prime(bits=512):

    return sympy.randprime(2\*\*(bits-1), 2\*\*bits)

def compute\_keys():

    p = generate\_prime()

    q = generate\_prime()

    n = p \* q

    phi\_n = (p - 1) \* (q - 1)

    e = 65537  #*public exponent*

    d=pow(e,-1,phi\_n) #*private exponent*

    return ((e, n), (d, n))

def encrypt(message, public\_key):

    e, n = public\_key

    message\_int=int.from\_bytes(message.encode(), 'big')

    cipher\_text=pow(message\_int,e,n)

    return cipher\_text

def decrypt(cipher\_text, private\_key):

    d, n = private\_key

    decrypted\_int=pow(cipher\_text,d,n)

    decrypted\_message=decrypted\_int.to\_bytes((decrypted\_int.bit\_length()+7)//8, 'big').decode()

    return decrypted\_message

if \_\_name\_\_=='\_\_main\_\_':

    print("Rivest–Shamir–Adleman (RSA) implementation")

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

    print(f"public key: {public\_key}")

    print(f"private key: {private\_key}")

    msg=input("Enter a message to encrypt:")

    print(f"Original message: {msg}")

    cipher\_text=encrypt(msg,public\_key)

    print(f"Encrypted Message: {cipher\_text}")

    decrypt\_message=decrypt(cipher\_text,private\_key)

    print(f"Decrypted Message: {decrypt\_message}")

Output:

