Assignment B4

Roll No: 41449

Title: Implementation of RSA

Problem Statement : Implementation of RSA

Objective : To understand how RSA algorithm works

Outcome : Understaning and implementation of asymmetric encryption using RSA.
Concept related theory :

RSA algorithm involves three steps
1. Key Generation
2. Encryption
3. Decryption

1. Key Generation

The key generation algorithm works as follows:
1. Generate two large random primes, p and q, of approximately equal size such that their
product $n=pq$ is of the required bit length, e.g. 1024 bits.
2. Compute $n=pq$ and $\phi=(p-1)()(q-1)()$
3. Choose an integer e, $1)(<e<\phi$, such that $\gcd(e,\phi)=1)$
4. Compute the secret exponent d, $1)(<d<\phi$, such that $ed\equiv 1 \bmod \phi$
5. The public key is (n,e) and the private key (n,d). Keep all the values d, p, q and ϕ secret.

2. Encryption:

Sender A does the following:-
1. Obtains the recipient B's public key (n,e)
2. Represents the plaintext message as a positive integer M with $1<M<n$
3. Computes the ciphertext $C=M_e \bmod n$
4. Sends the ciphertext C *t*o B.

3. Decryption

Recipient B does the following:-
1. Uses his private key (n,d) to compute $m=C_d \bmod n$
2. Extracts the plaintext from the message representative m

Conclusion :

Successfully implemented RSA algorithm

Result :

jupyter RSA Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```python
if __name__ == '__main__':
    '''
    Detect if the script is being run directly by the user
    '''
    print ("RSA Encrypter/ Decrypter")
    p = int(input("Enter a prime number (17, 19, 23, etc): "))
    q = int(input("Enter another prime number (Not one you entered above): "))
    print ("Generating your public/private keypairs now . . .")
    public, private = generate_keypair(p, q)
    print ("Your public key is ", public ," and your private key is ", private)
    message = input("Enter a message to encrypt with your private key: ")
    encrypted_msg = encrypt(private, message)
    print ("Your encrypted message is: ")
    print (" ".join(map(lambda x: str(x), encrypted_msg)))
    print ("Decrypting message with public key ", public ," . . .")
    print ("Your message is:")
    print (decrypt(public, encrypted_msg))
```

```
RSA Encrypter/ Decrypter
Enter a prime number (17, 19, 23, etc): 17
Enter another prime number (Not one you entered above): 29
Generating your public/private keypairs now . . .
Your public key is  (247, 493)  and your private key is  (263, 493)
Enter a message to encrypt with your private key: pratiksha sancheti
Your encrypted message is:
430 330 466 261 164 384 463 128 466 450 463 466 49 278 128 356 261 164
Decrypting message with public key  (247, 493)  . . .
Your message is:
pratiksha sancheti
```