

## RSA Implementation:

RSA is a complex topic, but the algorithm for implementing it is relatively simple. Especially compared to the exhaustiveness of verifying and understanding its properties.

To determine your key pairs:

1. Find two primes (p & q) randomly, defining 'n' as  $n = p * q$
2. Define  $\phi(n) = (p-1) * (q-1)$
3. Choose e at random, but such that  $e < \phi(n)$ , and e &  $\phi(n)$  are coprime(they share no common factors, i.e:  $\text{GCD}(e, \phi(n)) = 1$  )
4. Define d as the MMI of e mod  $\phi(n)$ , such that  $e * d \equiv 1 \pmod{\phi(n)}$  (done with pow() )
5. Publicize (e, n) as the public key, keep (d, n) as your private key.

(Discard  $\phi(n)$ , p, q)

For use of the created keys, both ends of RSA(encryption and decryption) are forms of modular exponentiation, either ' $m^e \bmod n = c$ ' or ' $c^d \bmod n = m$ ', which are both very similar.

(note that 'n' is used in both processes, while 'e' and 'd' are exclusive and inverses of each other)

Encryption:

Given: 'm'(an integer), (e, n) (the public key of the pair)

Return: 'c', a ciphered integer defined as  $c = (m^e) \bmod n$

Decryption:

Given: 'c'(a ciphered integer formed from the public key), (d, n) (the private key of the pair)

Return: 'm',

Your task is to implement those properties into a single, cohesive code that performs RSA encryption, such that it is abstracted as someone desiring to use RSA would prefer, instead of having to manually implement it every time. There should be a key generation, encryption, and decryption method.

- 1) Implement recursive gcd function
- 2) Implement Inverse Module function
- 3) Implement generateKeys function
- 4) Implement encrypt and decrypt function

SUBMIT : YOUR CODE AND Check if your code producing the encrypted and decrypted message for the numbers below:  $p=3$ ,  $q=11$  and  $e=7$ ,  $d=3$  and  $m=2$ ..