RSA ENCRYPTION AND DECRYPTION

|  |  |  |
| --- | --- | --- |
| Name | Section | BN |
| Menna Allah Ahmed | 2 | 30 |
| Nada Elsayed Mohammed | 2 | 33 |

1. Discuss the results you obtain in mathematical attack?

This attack depends on try all possible prime numbers, then the time will grow as the public key is growing

1. Justify why it happens?

From the inserted code snippet below, the idea the mathematical attack based on is trying to figure out the p, q. In other words, trying to factorize the n to find p, q.

We managed to get the p and q, now the attacker has e, n, p and q he can easily find phi n and obtain d “private key”.

But if the size of n is huge, the factorization process takes much more time. The greater the size of n is, the more difficult for the attacker to obtain p, q is reasonable time.

def mathematical\_attack (cipher, e, n):

    deciphered = ''

    for p in range(2, int(math.sqrt(n)+1)):

        if(n % p == 0):

            bob.q = n//p

            bob.e = e

            bob.p = p

            bob.n = n

            deciphered = bob.decrypt(cipher)

    return deciphered

for CCA:

it much faster than the brute force method because the idea of this attack is:

the attacker pretended to be the sender and sends to Bob, a message which is modified version from the cipher message he intercepted

C’ = C \* r e  mod n

When he sends back to bob, bob will decrypt the message by his private key d

Cd \* red = Med \* red since ed are inverse modulo n

He got M \* r, since r is unknown for the attacker, he will obtain the M without knowing the private key d.

So, it doesn’t depend on finding neither the p and q nor the d to decrypt the cipher message.

def CCA(C,e,n):

    r = ut.generate\_r(n)

    C\_dash= C \* ut.PowMod(r,e,n)

    Y = ut.str2int(bob.decrypt(ut.int2str(C\_dash)))

    M = ut.PowMod(Y \* (ut.InvertModulo(r,n)),1,n)

    print(ut.int2str(M))

    return(ut.int2str(M))