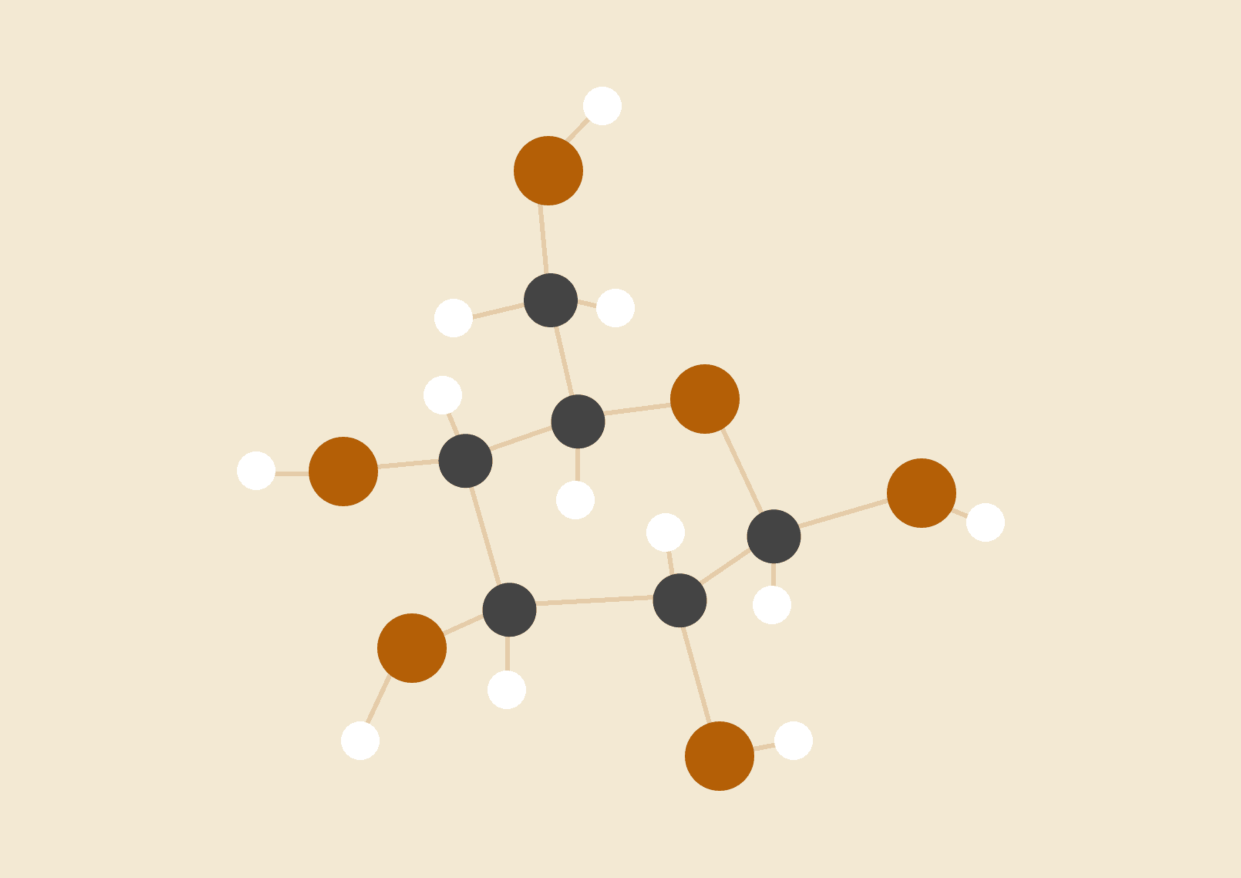
Computer Networks Lab 11



Alisetti Sai Vamsi

27/11/2021

111801002

1. Generating RSA public and private keys:

Program Name: Q1.py

Invocation: python Q1.py

Algorithm Outline:

1. **Generate two large primes using miller-rabin primality tests -> p,q**
2. **Calculating modulus for the keys -> n =p*q**
3. **Calculating the totient function -> totient_fn = (p-1) * (q-1)**
4. **Calculate coprime to totient function -> d = first number d that has gcd(d, totient_fn) = 1.**
5. **Calculate modular multiplicative inverse -> e = find the number e such that (d*e) mod totient_fn = 1**
6. **Public key = (e,n)**
7. **Private key = (d,n)**

2. Encrypting and signing using public and private keys:

Program Name: Q2.py

Invocation: python Q2.py

Algorithm Outline:

1. **Read public key of B and private key of A**
2. **Maximum size is set to n/32 where n is the number of bits of the key since if the plain text integer exceeds the value of n’ (value of n from the public key of B) then it cannot capture the cipher text and will produce repeated reminders because of the modulus taken by n.**
3. **Read the text from message.txt**
4. **Modify the text to accommodate the public key of B, which is used for verification on the receiver side.**
5. **Break the text into chunks based on this maximum size.**
6. **Open secret.txt**
7. **Loop through the chunked text**
    a. **Convert current text in loop context into integer**
    b. **Sign this integer with private key of A**
    c. **Encode the signed text with the public key of B**
    d. **Write it to the secret.txt file with space as delimiter**

3. Decryption using public and private keys:

Program Name: Q3.py

Invocation: python Q3.py

Algorithm Outline:

1. **Read public key of A, B and private key of B**
2. **Open secret.txt to read the encrypted data**
3. **For each chunk of integer in the encrypted data**
    a. **V = Decode the integer using private key of B**
    b. **Decode V using public key of A to undo the signature**
    c. **Convert the final integer into a string and append it to a running string**
4. **Check the message for public key of B**
    a. **If the message contains the public key of B then**
        i. **It came from A, and hence print the message**
    b. **Otherwise**
        i. **It came from someone else**
        ii. **Invalid message**
5. **If there are any errors in decoding then that means that the prime numbers generated are not exactly prime numbers.**