**Module 1 – RSA Encryption**

RSA is an asymmetric cryptographic algorithm used by to encrypt and decrypt messages, It uses two different keys hence the asymmetry.

* **Algorithm steps**

1.select to random prime numbers p,q (from the user)

2.calculate RSA modulus n = p\*q

3.calculate totient = (p-1) (q-1)

4.select public key e (Finding the first number that is co-prime with the totient)

5.select private key private d (using the inverse mod operation)

* **Input**
* p
* q
* message to be encrypted
* **output**

decrypted message

**Module 2 – sender, receiver**

In this module the receiver generates the keys needed for encryption and decryption randomly then it communicates the public key to the sender through a .txt file named ‘public.txt’

the sender reads the public key and encrypts the message then uses another .txt file to send the message named ‘message.txt’ meanwhile the receiver is waiting for the existence of ‘message.txt’ to read the sent message and decrypts it using the private key.

* **Receiver**
* Generate keys
* Write public key to file
* Wait for sender
* **Sender**
* Read public key if exists else wait
* After reading encrypt message
* Send message

**Module 3 – Encryption time**

The encryption process is ((char \*\* key) % n) for each character in the message. So, it depends on e and n.

The following two figures is the encryption time of a 429 character message with an increasing values of p,q

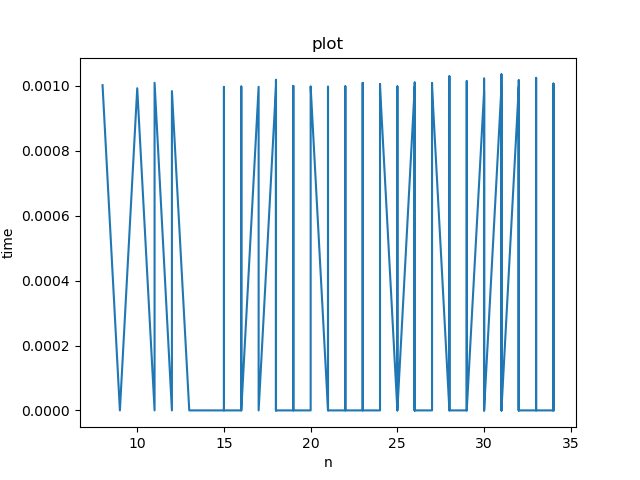


Figure 1 time vs values of n

As illustrated in figure 1, the time of encryption varies from 0 to 0.0010 i.e. it’s almost constant irrelevant of n.

This can be explained by the operation of finding (e), e is the first number that is co-prime with totient and smaller than it which means that (e) is almost always very small hence the small encryption time.

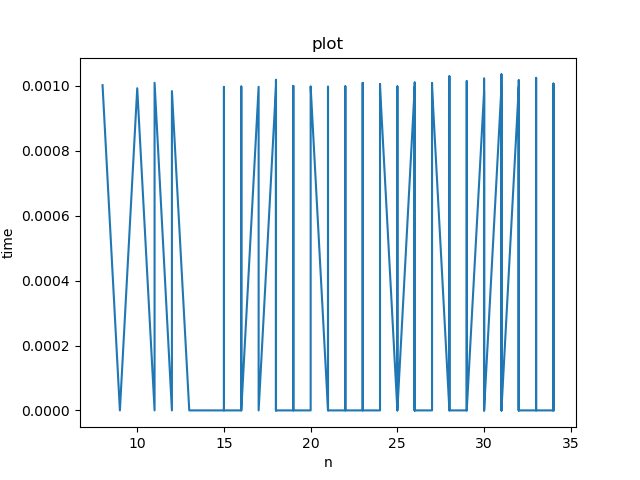


Figure 2 time vs number of bits of n

**Module 4 – RSA Attack**

In this module, implementation of a brute force attack on the RSA encryption algorithm.

Finding the decryption key by repeating the decryption process with different keys and comparing with original message.

The following two figures illustrate the decryption process against increasing values of n

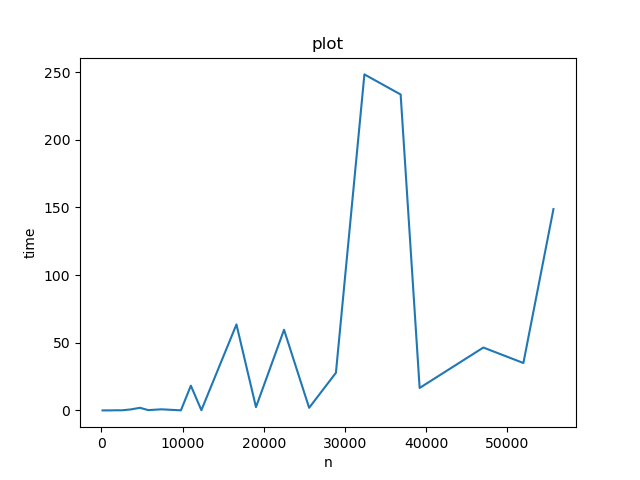


Figure 3 time vs values of n

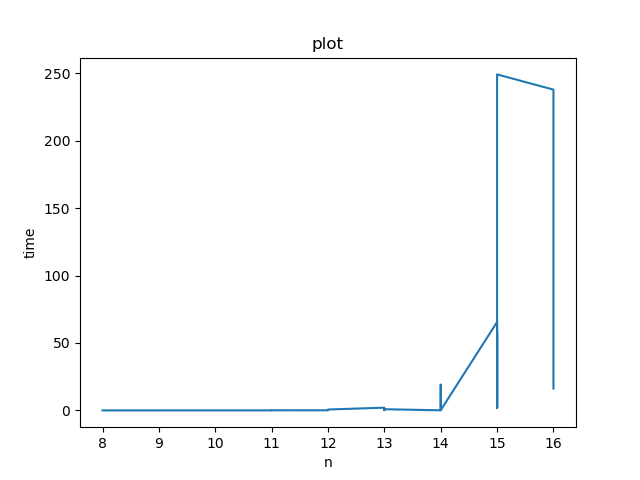


Figure 4 time vs number of bits of n