Computer Engineering Dept.
Faculty of Engineering
Cairo University **Cryptography Project**

Osama Magdy Aied 115

Mohammed Khaled Mohammed 2

Requirement 1:

Implementing the RSA algorithm was through Implementing 3 functions:

1 - RSA(num_bits) which takes the number of bits as an argument to determine the number of bits for the used key. Then it generates 2 big prime numbers which will be multiplied together to form n

The RSA algorithm is based on choosing 2 big prime numbers and calculating their product (n), the notion of their product (phi(n)). Then you are required to choose another number (e) that satisfies the condition of gcd(e, phi(n)) =1. The final number that needs to be Computed is d = e^-1 mod n.

The key pair generated is

Pu = (e, n) - - > send to all users who want to encrypt their data

Pr = (d, n) - - > kept secret to be used in decryption.

2 - Encrypt(message, pu):

This function takes the message in an integer form (the size of the message is depending on the number of bits used to generate keys), raises it to the power of e modulo n, and returns the result as another big integer

3 - Decrypt(cipher, pr):

This function takes the ciphertext as a big integer and raises it to the power of d modulo n. Which will return it to the original message integer. The returned integer can then be recovered in the string form to retrieve the original message.

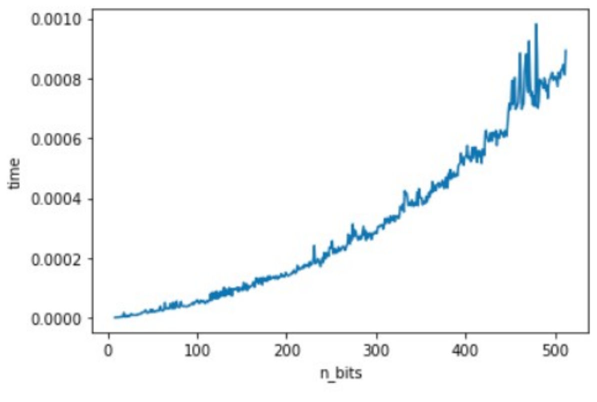
Requirement 2: Sender - Receiver application using socket programming in Python.

- At the receiver side (he is the one who generates the keys): we follow these steps
  a) first, import the socket library to be used in communication
  b) After importing, you need to create a socket object
  c) Choose the port to start communicating on (the IP, in this case, will be your localhost)
  d) Bind the port to the receiver socket
  e) Start listening to one client
  f) Generate your public and private keys to be used for communication
  g) Wait for the sender to request connections
  h) Start by sending the public key to the sender so he can text you
  i) As long as you didn't receive '0' as the message, keep reading
  j) Waiting for a message
     1- decode the received string
     2- convert it to an integer so it can be decrypted
     3- decrypt it is using your private key
     4- The first received character is the message length
     5- loop over it and start receiving characters one by one

- At the receiver side (he is the one who generates the keys):
  a) first, import the socket library to be used in communication
  b) After importing, you need to create a socket object
  c) Choose the port to start communicating on (the IP, in this case, will be your localhost)
  d) Connect to the port where the receiver is listening
  e) get the public key to encrypt messages
  f) waiting for messages to be typed
    1- To be general with different key sizes, we encrypt the message character by character
    2- Encrypt using the public key
    3- first send the length to the receiver (also encrypted)
    4- Send the message to the receiver character by character

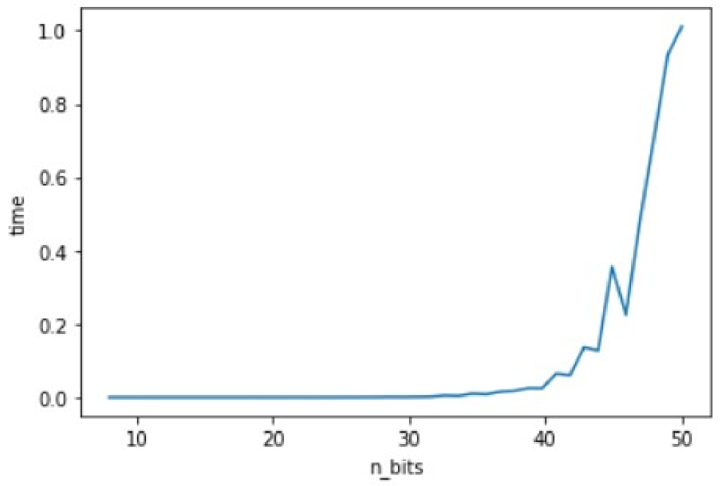
Requirement 3:

1)converted message to intger to be abloe to cypher it easier
2)generated keys with n_bits from 8 to 512 bits
3)calculated the time to encrypt message 100 times and averaged the time to get a smoother curve
4)plotted the curve using matplotlib



Requirement 4:

1) To break TSA given a public key, loop unitl n/2 and find a number that divides n
2)then the we get q as q=n//p
3)generate keys to break in a loop with size from 8 bits to 50 bits
4)break the RSA keys and measure the time, then plot it using matplotlib



Requirement 5:

Chosen Ciphertext attack

We have Alice who sends a message "M" to Bob encrypted with his public key (e,n). The encrypted message is C = M^e mod N. Eve intercepts C but can not decrypt it. So, Eve multiplies C by a random number r raised to the power e and all modulo N. C_eve = C * r^e mod N. Then he sends it to Bob who decrypts it to C_eve^d mod N. Eve can now recover the original message M by multiplying the result from Bob by r^-1 mod N. (r ^ -1)*(C_eve ^ d) Ξ (r ^-1) * ((C * (r^e))^d) Ξ (r ^-1) * (C^d) * (r^ed) Ξ (r ^ -1) * M * r Ξ M mod N.

a) Following those steps:
   First, generate the RSA at Bob's side
b) Alice wants to send A message
c) Eve can see the encrypted cipher

d) Eve selects a random number
e) Eve encrypts r with the public key
f) Eve multiplies it by cipher mod N
g) Eve sends cipher_2 to Bob who decrypts it and sends her what do you mean by this "result"
h) Then Eve multiplies r_inverse by cipher_2 to get the message