Assignment 6: Public Key Cryptography

Nelly Sin

November 2021

1 Task

Task one that generates keys (private and public) encrypt files decrypt files

GMP library: We must install the package

Global random state variable called state, generating large random numbers. Using the random seed for the state. We also must clear it.

2 Randstate

This initialize the global random state named ”state” and the random seed. We will be using this throughout the assignment.

3 Numtheory

Modular exponentiation using the binary squaring method. We must translate it to gmp: mpzt

3.1 Power-Mod(a,d,n)

This is a fast modular exponentiation, computing base raised to the ”exponent” power modulo, ”modulus” and storing the computed result in out.

v = 1
p = a
while d is greater than 0
check if d is odd
if d is odd, assign v to (v x p)
p will equal to (p x p) mod n
d will equal to (d/2)

3.2 Primality testing:

Because these are very large numbers we must use a randomized number to tell us if something is prime. We will using Miller-Rabin Primality testing

Given integer "n" we choose a positive integer a ¡ n. Letting

$$2^s d = n - 1 \tag{1}$$

create a loop such that d must be odd
for range 1 to k
choose random positive integer with of an array withn - 2
y = POWER-MOD(a,d,n)
if y is not 1 or y is not n - 1
j = 1
while j ¡ s - 1 and y != n -1
y = POWER-MOD(y, 2, n)
if y == 1
return false
j = j + 1
y != n -1
return false
return true

3.2.1 Is Prime

This is part of the Miller-Rabin primality test to indicate whether n is prime or not using the "iters" of the Miller-Rabin iteration. Whe we create the two large prime numbers of p and q it verifies whether or not the integer is prime.

3.2.2 Make Prime

This function generates a new prime number stored in p. The prime that is generated must be tested in isprime using "iters".

3.3 Modular Inverses

We will be using Euclid's algorithm, which is an efficient method to compute the greatest common divisor of two integers, the largest number that can divide them both with a remainder of 0. We must implement a function to compute the GCD, greatest common divider.

3.4 Greatest Common Divider

GCD(a,b)
while b != 0
t = b
b = a mod b
a = t
return a

3.4.1 Mod-Inverse

When we use mod-inverse, this will be an essential part of decrypting of the keys.

(r, r‘) = (n, a)
(t, t‘) = (0,1)
while r‘ != 0
q = floor(r / r‘)
rtemp = r
r = r‘
r‘ = rtemp - q x r‘
ttemp = t
t = t‘
t‘ = ttemp - q x t‘
if r > 1
return no inverse
if r < 0
t = t + n
return t

4 RSA

RSA Coprime: a and b are coprime if gcd(a,b) = 1 relatively prime

Alice: primes p and q
n = pq <= very large
toshent(n) = (p-1)(q-1)
e = 65537

calculate the inverse of e
d = e-1 mod toshient(n)
de = 1 mod toshient(n)
r-1 mod 13 = 8 mod 13
5* 8 = 40 mod 13
= 1 and 13

d is your private key, n and e is the public key

E(m) = me mod n
D(E(m)) = (E(m))d mod n

make public key, n bits = how many bits go into the public key. given range encrypt files: we must encrypt the bytes. we want to read in a block size of a byte, we want to convert it to a large integer and encrypt the large integer.

$m < n$

rsa decrypt is similar

rsa sign when trying to encrypt the message.

given sudo code:
write = n = 2sr + 1 (r is odd) ¡- loop to make sure it's odd

We want to compute s and r, and r should end up odd.
while(is - even(r))
do something
is prime: k is a number of iterations make prime: