Assignment 5: Public Key Cryptography

Kelly Liu

November 11, 2021

1 Description

In this assignment, we are making a RSA public key system where we are making a key generator, encryptor and decryptor program. We will make a public key to encrypt a file and then use the private key to decrypt the file. We will also be working with GNU Multiple Precision Arithmetic Library while creating some of our own math functions in order to create our RSA system.

2 Pseudocode

2.1 randstate.c

Include all files Creating an global variable state

2.1.1 randstate_init

initialize state
set our state to seed

2.1.2 randstate_clear

clear state

2.2 numtheory.c

Include all files

2.2.1 gcd

Create an zero,tempb and tempa variable
Intialize all variables
Set tempb to b and tempa to a

Create an while loop that runs while temp is not equal to 0
Set a temp variable to tempb
Set tempb to tempa%tempb
Set tempa to temp
Out of while loop
Set output variable to tempa
Clear all variables

2.2.2 mod_inverse

Create an r1,r2,t1,t2,zero,final,one variable
Initialize all variables
Set r1 to n r2 to a
Create an while loop to run while r2 is not 0
Create an q variable and set that to r1 divided by r2
Initialize a temp variable that will hold our r2 variable
Set r1 to r1 minus q time r2
Set r2 to r1 and r1 to temp
Set our temp variable that will now hold our t2 variable
Set t1 to t1 minus q time t2
Set t2 to t1 and t1 to temp
Out of while loop
Create an if statement to check if r is bigger than 1
If true, set output to zero to say there's no inverse
Create an if statement to check if t is smaller than 0
If so, set output to t2 plus n
Set output to t1
Clear all variables

2.2.3 pow_mod

Create a v,p,zero,two,temp,tempe variable
Initialize all variables
Set p to base, tempe to exponent
While tempe is biggier than 0
Set temp to tempe % 2
Create an if statement to check if it's odd
If so, set v to v times p then v% modulus
Set p equal to p times p
Set d to the floor division of 2
Set our out variable to v
Clear all variables

2.2.4 is_prime

Create a s,r,y,j,jcomp,temp,ncomp,temp,zero,one,two,three,ntemp,a,mthree,stemp variables
Initialize all variables
Set ntemp to n - 1
Create an if statement to check if n is 0
If so return false and clear all variables
Create an if statement to check if n is 1 or 2 or 3
If so return true and clear all variables
Create an if statement to check if it's even
If so return false and clear all variables
Create an while loop to run while temp equals 0 (Outter while loop)
Increment s by one
Get the remainder of the division of ntemp and s and set that equal to temp
Out of while loop
Decrement s by one
Set stemp to s
Set r to the quotient of ntemp and s
Use a for loop to iterate from one to iters (including iters)
Set mthree to n minus three
Get an random variable a from zero to mthree
Increment a by two
Pow mod using our earlier function a,r,n and set it to y
Set jcomp to s minus one
Set ncomp to n minus one
Create an if statement to check if y isnt one and y isnt ncomp (Outter if statement)
Set j to one
Create an while loop to run while j is smaller than jcomp and y isnt ncomp(Inner while loop)
Pow mod using our earlier function y,r,n and set it to y
Create an if statement to check if y equals 1
If so, return false and clear all variables
Increment j by one
Create an if statement to check if y isn't ncomp
If so, return false and clear all variables
End of inner while loop
End of outter if statement
End of outter while loop
Clear all variables and return true

2.2.5 make_prime

Create an random and temp variable
Initialize all variables
While my random variable isn't prime (Outter while loop)
Create an new random variable
Set temp to random
Create a size variable that is equal to the size of my random variable in base 2
While my size is not in our range (Inner while loop)
Create a new random variable
Set temp to random
Set size variable to the size of my random variable in base 2
End of inner while loop
End of outter while loop
Set output to the prime number we made
Clear all variables

2.3 decrypt.c

I will use getopt for my switch cases
Open the private key using fopen while testing for errors
I will read the public key and print our verbose output if prompted
Decrpy the file using the private key
close the private key and any variables.

2.4 encrypt.c

I will use getopt for my switch cases
Open the public key using fopen while testing for errors
I will read the public key and print our verbose output if prompted
Convert the username and verify the signature while testing for errors
Encrypt the file and close the public key and any variables

2.5 keygen.c

I will use getopt for my switch cases.
Then I will open my files using fopen while testing for errors.
I will then use fchmod and fileno to set the private key permissions and other permissions.
I will then set the randomm seed.
Create my keys.
Get the user's names and write the keys.
Then I will close my key files and clear my random state and variables.

2.6 rsa.c

2.6.1 make_pub

Create my pupper and lower bound variables
Create pbits using (random() mod (upper minus lower) plus 1)
Create qbits and set it to nbits minus pbits
Increment nbits and qbits by one
Use make prime to make p and q
Multiply my n argument by p and q
Create a one,temp,totient,ptemp,qtemp variable
Initialize all variables
Set ptemp to p and qtemp to q
Subtract ptemp and qtemp by one
Set my totient to qtemp times ptemp
Get a random variable and set it to e
Set temp to the gcd of e and totient
Create an while loop to run while our gcd isn't our coprime
Create another random e and set temp to the new gcd of e and totient
End of while loop
Clear all variables

2.6.2 rsa_write_pub

Print out n,e,s, and username to our pbfile

2.6.3 rsa_read_pub

Use fscanf to read each line of the file and set it to it's corresponding argument

2.6.4 rsa_make_priv

Create an one,totient,ptemp,qtemp variables Initialize all variables Set ptemp to p and qtemp to q Subtract ptemp and qtemp by one Set the totient to qtemp times ptemp Set the output to the mod inverse of e and totient Clear all variables

2.6.5 rsa_write_priv

Print out n,d to our pvfile

2.6.6 rsa_read_priv

Use fscanf to read each line of the file and set it to it's corresponding argument

2.6.7 rsa_encrypt

Set our c to the pow mod of m, e and n

2.6.8 rsa_encrypt_file

Create a m variable and initialize it
Create our k variable and set it equal to the formula provided
Create our block array and initialize it
Set the initial element in our block array to 0xFF
Create an while loop to run until there's no more bits
Convert the bytes read using mpz_import
Encrypt our message using m,e,n
Print our message to our outfile
End of while statement
Free our block array
Clear m

2.6.9 rsa_decrypt

Set our m to the pow mod of c,d,n

2.6.10 rsa_encrypt_file

Create a m variable and initialize it
Create our k variable and set it equal to the formula provided
Create our block array and initialize it
Create an while loop to run until there's no more bits
Decrypt our message using m,d,n setting it to m
Convert the bytes read using mpz_export
Fwrite our message to our outfile
End of while statement
Free our block array
Clear m

2.6.11 rsa_sign

Set s to the powmod of m,d,n

2.6.12 rsa_verify

Create a t variable and initialize it
Set the t variable to the pow mod of s,e,n

Check to see if the m passed in from our argument is equal to t
And if it is, clear t and return true
If not, clear t and return false

3 Files

numtheory.c- A source file for implementing of my functions.
numtheory.h- A header file that specifies the interface for numtheory.c.
randstate.c- A source file for implementing the random state interface of my RSA and numtheory functions.
randstate.h- A header file that resets and initializes my random state.
rsa.c- A source file for implementing my RSA library.
rsa.h- A header file that specifies the interface for rsa.c.
encrypt.c- A source file for my encrypt program.
decrypt.c- A source file for my decrypt program.
keygen.c- A source file for my keygen program.
Makefile- This allows us to use clang and compile our program.
README.md- In markdown format, it tells us how to run the program and how the program was made.
DESIGN.pdf- This is how I started thinking about how to code the program.

4 Credit

1. Professor Long has provided pseudocode in the Assignment description PDF for our files as well as describing specific things that we should be doing.
2. Professor Long has provided us with a few files in the resources folder in Assignment 6.

5 Errors

1. There were alot of parts where I was super careless because I didn't read the assignment document carefully enough.
2. In numtheory, I had to make sure that I didn't change any of the arguments passed in directly. I fixed this by creating temporary variables that held the arguments.
3. In numtheory, I had issues with the division functions in the GMP library because the one I was using takes in an mp_bitcnt_t. This was fixed once I realized what it was.
4. I had to add some test cases to is_prime because the method we use isn't completely accurate.
5. My keygen was erroring out because I had set my username to a base 0 when

it should be base 62.
6. My rsa make pub was erroring out because I had flipped my upper and lower bound when I was creating my random number.
7. I forgot to clear some variables after I returned stuff.
8. I had a test case for my is_prime that was incorrect because I was being stupid. (Test case 1).