ASSIGNMENT 5 DESIGN DOCUMENT

Pehara Vidangamachchi

October 28th 2022

Ethan L. Miller

About:

This assignment aims to create working programs for keygen, randstate, encrypt, numtheory, decrypt and rsa. These programs all combine together to create the finished product for an encoder and decoder using the RSA standard __. Keygen is responsible for creating a random key for encrypt and decrypt, numtheory helps create the mathematical process behind the encryption and decryption of the program, followed by randstate which assists all the programs by generating randomness. Encrypt will encrypt a file with a public key and Decrypt will decrypt the file using its private key. The assignment at hand tasks us with the creation of our own implementation of this function.

Design Process for decrypt.c

The purpose of this program is to create a function decrypt function that is able to through several parse-options. It is known as the process of decoding encoded data. Decrypt is a function used to revert an encrypted message back into something readable without sacrificing safety. This is done with the use of keys, private and public, which assist with the process. For decryption, the private key is used to translate the message into something readable once again. I aim to achieve this by using a multitude of for and if statements alongside the use of the GMP library to help create a functioning program.

#include any header files

 # define the options, i, o, n, v and h.

Create the main function for decryption:

Initialize the variables used within the function with, int, char and mpz_t

Be sure to set the files to file types

Create a while loop to process the user input:

Create switch cases to parse through each option:

Using case declare the case

Attach the cases matching component

Break, and repeat for all the cases

Set the temp variable equal to the actual variable and open the file

Create a function to open the input file and check for errors in the input file

Print a statement in case the input is nonexistent

Set the temp variable equal to the actual variable and open the file

Create a function that checks the output file and checks for errors

Print a statement in the case that the file does not exist

Set the temp variable equal to the actual variable and open the file

Create a function to open the file key and check for errors in the private key

Print a statement in the case that the input is out of range

Use the RSA read pub function to read the private key from the file

Create a segment for the verbose output and print out the contents using an if Statement:

Using print statements, print out all the contents on each line

Create an if statement for the case that the signature is wrong and doesn't match

Print statement to let the user know the issue

Using the RSA encrypt file function, encrypt the file

Clear all the mpz variables

Close all the files

Return 0

Design Process for encrypt.c

The purpose of this program is to create an encrypt function that is able to encrypt a message and additionally parse through several different options. Encryption is the process of encoding a given set of data and converting it to what's known as cypher text. Encryption is a function that takes a message and encrypts it with a public key, so it's only readable if a private key is used to decrypt it. With the assistance of keys, the program is able to create an encrypted function alongside a mathematical formula to help translate its method of encryption. In order to achieve this I used the GMP library and its multitude of features, followed by several if, switch, while and case statements. When using the mpz_t function, I had to be careful to initialize all my

variables as well as clear their memory at the end of the programs o they wouldn't use too much space.

#include any header files

 # define the options, i, o, n, v and h.

Create the main function:

Initialize the variables used within the function with, int, char and mpz_t

Be sure to set the files to file types

Make it so that the username generates and holds a certain number of memory

Create a while loop to process the user input:

Create switch cases to parse through each option:

Using case declare the case

Attach the cases matching component

Break, and repeat for all the cases

Set the temp variable equal to the actual variable and open the file

Create a function to open the file key and check for errors in the input file

Print a statement in case the input is nonexistent

Set the temp variable equal to the real variable and open the file

Create a function that checks the output file and checks for errors

Print a statement in the case that the file does not exist

Set the temp variable equal to the real variable and open the file

Create a function to open the file key and check for errors in the public key

Print a statement in the case that the input is out of range

Use the RSA make pub function to read the public key from the file

Create a segment for the verbose output and print out the contents using an if Statement:

Using print statements, print out all the contents on each line

Create an if statement for the case that the signature is wrong and doesn't match

Print statement to let the user know the issue

Using the RSA encrypt file function, encrypt the file

Clear all the mpz variables

Free the username of the memory it's using up

Close all the files

Return 0

Design Process for numtheory.c

This program serves the purpose of containing the correct implementation of the number theory function. This will contain functions such as modular exponentiation and modular inverses which will help with the math behind encryption, decryption and key generation. Following what's given in the assignment, I can use the given pseudocode to translate that over to my own interpretation of the pseudocode.

#include any header files

#define any variables that'll be used

Create a make prime function:

Set it so that a random prime number generates

Create a while loop that iters through as long as the input is not prime:

When it is not prime use a random mpz function allocates its parts

Create a power mod function:

Initialize and declare all your variables and values using the GMP library

Using the GMP library set it so that v is equal to one and p is equal to a

Create a temp variable

Create a while loop so when d is greater than zero:

*Make it so that if d is odd it will set c = v * mod n*

*Then using the GMP library make it so that p * p * mod n*

Make sure to return and clear memory

Create an is prime function:

Using the miller Rabin formula execute the function

Create a segment of the function properly to make sure that s remains at the right spot and that r is odd

This is done with an if statement that checks whether n = 2 or n = 3:

Which returns true

Otherwise using an elsif statement when n is even or n is less than 3:

Return false because a negative number is not a prime number

Using the GMP library make it so that mf is equal to n

And then mf is equal to n minus one, which then should be divided by two and

Given a temp value where it is equal to mf and the mod of 2

Then create a while loop so that when temp is equal to i

It'll divide r by 2, and then again but set it equal to temp and finally set

Set s equal to s minus one

Once this is all done then you may continue with the rest of the function by

Placing it in a for loop:

Using the GMP library to declare, follow it by doing the math

Create a random variable that goes from zero to the size limit minus one

Then set that random variable equal to itself minus two

Followed by setting it all equal and modding it

Then create a new if statement with a given range:

Which contains more supporting math where sj = s - 1

Create a while loop if j is less than sj and y doesn't equal mf:

Make y = powmod of y, 2 and n

Followed by an if statement so that when y = 1:

It will clear all the memory and return false

Otherwise, it will make j = to j + 1

Create an if statement so when y doesn't equal mf:

It will clear all the memory and return false

Clear any other memory used and return true at the end

Create a function for gcd:

Create a while loop so that when b is not equal to zero it runs

Initialise all the important variables using the GMP library

Set t = to a mod b

Compute the math using the GMP library and clear memory

Return the results

Create a modular exponentiation function:

Initialize any variables using the GMP library

Make r = and r prime = a

Create a while loop for when r does not equal zero:

Then compute the math portion that sets t = to b, b = a mod b and a = t

At the end of the loop clear the memory of all the mpz functions

Create an if statement so in the scenario taht r is greater than one:

Itll return no inverse and clear all the memory.

Followed by an if statement so that if t is greater than 1:

T will have itself added by n onto it.

Finally, return and clear all the GMP values.

Design Process for randstate.c

This file contains the implementation of the random state interface used for the RSA library and the number theory functions for my code. This file is essential to using RSA public and private keys. This file helps bring all the necessary code into one compilable program.

#include any header files

Initialize state with the gmp library

Create a function that generates the seed:

Using GMP functions rand init and randseed

Create another function the clear the memory within randstate

Design Process for rsa.c

This program will contain the implementation of the RSA library that will be used within my program. The file RSA.c is the library for RSA that I can utilize within my code. This helps to specify certain command line options, such as the minimum bits and the number of miller Rabin iterations.

#include any header files

 # Initialize or define any important variables

Create an lcm function:

Using gcd you can calculate the lcm just by formulating the absolute value of

*p * q divided by r*

Clear memory

Create a function for rsa makes pub which generates a public key:

Initialize any variables using the GMP library

Use math functions from numtheory such as make prime with the input

Calculate the lcm = (p - 1)(q - 1)

Create a while loop so when the lcm does not equal one it runs:

Generate a random number within this loop using the GMP library.

Clear memory at the end of the loop

Set n = to pe and lambda equal to e at the end before returning and clearing Memory

Create a make priv function:

Print out the contents of each variable

Create a read pub function:

Print out the contents of each variable

Create a write private function:

Print out the contents of each variable

Create a read print function:

Print out the contents of each variable

Create a rsa encrypt function:

Using power mod set the contents equal to their respective counterpart

Create an encrypt file function:

Using the GMP library to declare any variables

Create an if statement in the case that n is greater than one:

Which then finds the log of n - 1 over 8

Allocate memory to the array block

Set the zeroth byte

Create a while loop so when c is not equal to the end of the file:

It will convert the segment and encrypt the file

Free the memory being used and clear the rest

Create a decrypt function:

Decrypt the file by getting the power mod

Clear any memory being used

Create a decrypt function:

Initialize any variables using the GMP library

Create an if statement so when n is greater than one:

It finds the log of n - 1 over 8

Allocate memory to the array block

Create a while loop to scan the file when it is not equal to the end of the file:

Find the log of n - 1 over 8 again

The convert by running decrypt

Export the results and write out the file

Create a for statement to set the array size back to empty at the end

Free any memory used or clear it

Create a function called sign:

This gets the sign by using power mod and setting it eqaul to a

Clear any memory used

Create a function called verify:

This checks for the files completion using power mod

Create an if statement so when a is equal to b

Then it will return true

Or else it return false

And finally clear all the memory

Design Process for keygen.c

This program will contain the implementation of the RSA library that will be used within my program. Using this program helps create keys that will be utilized within other segments of the code that basically creates a random key to helping encrypt and decrypt the sequence. This is done through several supporting functions from numtheory and RSA. For this file, I had to use the GMP library for multiple different functions. I mainly utilized mpz type, alongside a multitude of whiles, switch, case, and if statements.

#include any header files

 # define the options, b, i, n, d, s, v, h.

Create the main function:

Initialize the variables used within the function with, int, char and mpz_t

Create a while loop to process the user input:

Create switch cases to parse through each option:

Using case declare the case

Attach the cases matching component

Break, and repeat for all the cases

Set the temp variable equal to the real variable and open the file

Create a function to open the file key and check for errors for public key

Print a statement in the case the input is outside the range

Set the temp variable equal to the real variable and open the file

Add fchmod and fileno for the public key

Set the file to be editable and in the case that the file does not exist print such

Set the temp variable equal to the real variable and open the file

Create a function to open the file key and check for errors for private key

Print a statement in the case that the input is out of range

Set the temp variable equal to the real variable and open the file

Add fchmod and fileno for the private key

Set the file to be editable and in the case that the file doesn't exist print such

Initiate the random state using the random seed

Make the RSA keys for private and public

Get the user's name using the environment

Convert that information using a mpz function and setting it to username

Compute the user's signature by using the RSA sign function

Then use the RSA private and public function to allocate the username to them

Now create a segment for the verbose output and print out the contents using an if Statement:

Using print statements, print out all the contents on each line

Use randstate clear to clear up the memory allocated to state

Clear all the mpz variables

Close all the files

Return 0