Design Document

Manikanta Illuri

The design and Ideas for this assignment were derived from TA and tutor help as well as some aspects from my previous quarters work on a similar assignment.

FINAL Design

Objective: The objective of this assignment is to create a program that can encrypt and decrypt a given input. For this a ss ignment I will be using keygen to produce an ss public and private key pair and use the encrypt and decrypt files to manipulate the input.

Deliverables:

- Keygen.c
- Encrypt.c
- Decrypt,c
- Numtheroy.c
- Numtheory.h
- Randstad.c
- Randstate.h
- ss .c
- ss .h
- Makefile
- DESIGN.pdf
- WRITEUP.pdf
- README.md

Num Theory

void pow_mod(mpz_t out, mpz_t base, mpz_t exponent, mpz_t modulus)

- Will initialize 6 variables 2 of which are temp variable by using mpz_t and mpz_inits
- Will set value of v to 1 and set p to a
- WIll initiate a while loop that will run while the ex value is greater than 0. This will be done using the mpz_cmp_ui function
    - Within the loop I will have an if statement that will check if the number is odd or not by dividing ex by 2 == 0.
        - Then will multiply v and p and store in vp
        - Then will vp and n and store in temp
        - Then will set v to temp
    - Outside the if will multiply p with p and store in temp
    - Then will mod ptemp and n
    - Then will divide e with 2 and store back in ex
- Outside the loop will set v to o a

- Then will clear all variables
-

- This function will perform modular exponentiation by computing the base raised to the exponent power modulo modulus and storing the value in an out var. I will be implementing a while loop that will check the value greater than 0. Within the loop I will use temp variables and a modulus operator to initiate the exponentiation. Outside the loop I will multiply the two temp values.

bool is_prime(mpz_t n, uint64_t iters)

- I will first check for any cases where the values are already prime such as 2 or 3 using an if logic
- For thai function I will initiate variables and set them to NULL
- Will then sub n1 with 1 and set back in n1()n - minus 1)
- Will set n to res(result)
- Will set 0 to val variable
- Will do mod of res and 2 and set in rm2 (r mod 2)
- Initiate a while loop that will check whether rm2 is not equal to 0 when compared to 1 -
  Within the while loop will do fdiv of res with 2 and store back in res
  - Will do mod of res 2 and store back in rm2
  - Will add val and 1 and store back in val
- Outside the while loop will subtract val with 1 and set back in s1(s-1)
- Will set pm2(p mod 2) to value 2 // doing so because powmod only takes mpz types, got help from TA
- Initiate a for loop that will set temp to 1 and cmp temp to iters using mpz cmp and will add 1 to temp and store in temp // that loop will loop until user enter value for number of iterations is reached.
  - Within in the loop will use the random to generate a random number and store it in a - Will add 2 to aval and store back in aval
  - Will call pow mod function and pass the values yval, aval, res, n
  - Will have a check to check if yval cmp 1 is not equal to 0 and yval cmp n minus 1 is not equal to 0
    - Will set j value to 1
    - Initiate a while loop that will run while j cmp s -1 is less than or equal to 0 and yval cmp n - 1 is not equal to 0
      - Will then call on pow_mod of yval and yval, pm2, n minus 1
    - Will have a check to check is yval cmp 1 is == 0
      - Which will return false
    - Then outside the cech it will add j to 1 and store back in j
    - Tbe will have another check to check if yval cmp n minus 1 is not equal to 0 - Which will return false
- Will clear all variables and return the value true as this function is a bool

- Thor function will indicate if a number is prime or not. To do so the function will have temp variables that are set to NULL. Then the variables will be sent to a while loop that checks if temp != 0. Within the loop an if will check if the values are divisible then it will use the modulus operation.

void make_prime(mpz_t p, uint64_t bits, uint64_t iters)

- Will initiate temp variable val and set it to NULL
- Then will have a variable of int type called tf that will store 0 and 1 values which will act as the conduction for the while loop later on.
- Then will do uipow_ui for 2 and bits and store in temp variable val
- Will initiate a whole loop that runs while tf is == 0
    - Will call mpz_randomb and pass state and bits and store in p
    - Then will add value and p and store back in p
    - Will have a check if is the p value is true and if so will change tf to 1 which will stop while loop late on
    - Will have to check if its prime return false and will change tf to 0 to continue checking. - Outside loop will clear val variabel
- This function is meant to generate prime numbers that are at least a bit length long. For this implementation I will initiate 2 temp variables that will be set to NULL. Then using a while loop I will check if the conviction while temp ==0, then it will call random and pass in n bits as an argument. Outside the loop it will clear the variables.

void gcd(mpz_t d, mpz_t a, mpz_t b)

- Will have an if statement what mpz_set the value 0 to d if the below cases occur: - The mpz_cmp_ui will check the parameters and will return 0 if b or a are equal to 0 or 1 - If b == 0
    - If b == 1
    - If a == 0
    - If a == 1
- I will create the temp variables and set them to NULL
    - Val, vala, valb
- Will set vala to a && valb to b
- Initiate a while loop that runs while the the b val is not equal to 0
    - Within while loop will set val to val
    - Then will perform mod of vala and valb and store in valb
    - Tnen will set val to vala
- Outside loop will set d which i the return value to vala
- Then will clear all temp variables with mpz_clears

- For the function I will be writing a program that computes the greatest common divisor of a and b and store them in temp variables. I will be using a loop that will check if temp is != 0 and will swap the values and mod them returning the other variable.

void mod_inverse(mpz_t i, mpz_t a, mpz_t n)

- Will create temp variables and will initialize them to NULL
    - R, rp, t, tp, q, qtp, qrp
- Will also initiate and set NULL to variable s:
    - Tempoval, and temo_val1
- Will use mpz_set to set n val to the temp variable r
- Will use mpz_set to set a value to temp variable rp
- Will use mpz_set_ui to set t to val 0
- Will use mpz_set_ui to set tp(tprime) to val 1
- Initiate while loop that runs while variabel rp (r prime) is not equal to 0
    - Within in while loop will use mpz_fdiv to divide r from rp and store value in q - Then will set tempval to r
    - Will use mpz_set to set rp to r
    - Will multiply rp with q val and store in qrp
    - Will use mpz_sub to sub tempval and qrp and store in rp
    - Will set t to tempval_1 variabel
    - Will set tp to t variable
    - Will do mpz_mul for q and tp and store in qtp
    - Will do subtraction of tempval_1 and qtp and store in tp
- Outside the while loop
- Will have a check to make sure r value is greater 0
    - Will set out value to 0
- Will have another check to check is t value is greater than 0
    - Will add t and n and store in val n
    - Will set out val to t
- Will use mpz_set to set out to t and will clear all values.

- This function is used to compute the inverse i of a module n. For the implementation of this function I will begin a while loop that will run while the temp variable is prime. And is not equal to 0. Within the while loop another nested loop will be present that will use the div function to divide the values of the two temp variables. Then it will return the values outside the loop.

<u>ss_Library</u>

void ss _make_pub(mpz_t p, mpz_t q, mpz_t n, uint64_t nbits, uint64_t iters)

- I will have a mpz_t variable called qminus 1 to store the value of q minus 1 and will initiate it
- Will create two uint64 bit variables for the bits and qubits
    - Pbits I will (random() % (((2 * nbits) / 5) - (nbits / 5) + 1)) + (nbits / 5);
    - Qbits I will subtract nbits with pbits an store in qbits

- Then will call my make prime function on both bits and qubits and pa ss in iters and store the output in p and q variables
- Then I will use mpz sub q -1 and set it to qminu 1
- While(mpz_cmp(q_minus_1, p) == 0 || mpz_cmp_ui(q_minus_1, 0) == 0)
    - I will call make prime on both qubits and bits again and set it to p and q values
    - Then I will sub the value of -1 and place it in q minus 1
- Outside the loop I will multiply p and p and n and q and place their values in n, then I will clear everything
- For the implementation of this function I will be having two large prime numbers and will compe n by doing p*p*q. For this function I plan on using a while loop that iterates while nbits is /5 is greater than 0. Inside the I will perform the multiplication operator and will set them the temp variable to the final value.

void ss _write_pub(mpz_t n, char username[], FILE *pbfile)

- For this function I will make a function that will write to the ssh key file. I will use the fscanf function to print to the output file.

void ss _read_pub(mpz_t n, char username[], FILE *pbfile)

- For this function I will fscanf

void ss _make_priv(mpz_t d, mpz_t p, mpz_t q)

- I will create variables and set them to NULL
- First I will set p-1 and q-1 by using the mpz sub
- Then I will multiply p and q and set it to pq
- Then I will multiply pq and q to n
- Then I will calculate the gcd of p min and q min and set it to a variable called gcd value
- Then I will multiply p-1 and q-1
- Then I will divide cgc valeua nd p-1 q-1 value and set them in lambda variable
- Then I will call mpz inverse
- Then I will learn the variables.

- For the make private using a set of temp variables and by setting them equal to NULL. I will also use the abs function to store them into the output file.

void ss _write_priv(mpz_t pq, mpz_t d, FILE *pvfile)

- I will havea ss key file that I will write to using the w permi ss ion.

void ss _read_priv(mpz_t pq, mpz_t d, FILE *pvfile)

- For the function I will use the ss h key output file and read the file by using the r permi ss ions.

void ss _encrypt(mpz_t c, mpz_t m, mpz_t n)

- I will call the pow mod function and pa ss in nece ss ary values

void ss _encrypt_file(FILE *infile, FILE *outfile, mpz_t n)

- Will initiate variables and set them to NULL
    - M, fnl_ency
- Will initiate k which is a variable that store the value of size base (n, 2)-1 /8 as a size_t type - Will use calloc to dynamically allocate an array. Using uint8_t
- As mentioned in the a ss ignment doc will do a[0] = 0xFF
- And set a new size_t variable s as explained in the a ss ignment doc (its to print a certain number of bits)
- Initiate a while conduction that will run till it is not the end of a file with the parameter in file - Will set the s variable to fread the infile
    - Will use mpz import to the order parameter of mpz_import() to 1 for the most significant wordfirst, 1 for the endian parameter, and 0 for the nails parameter
    - Then will call rsa_ecrypt and send the parameter m, e, n
    - Will use gmp fprintf to print the to the outfile
- Outside loop will clear all variables

- This function will use the contents of the infile and will encrypt them using modn and a while loop. And will check the below
- While there are still unproce ss ed bytes in infile: (a) Read at most k−1 bytes in from infile, and let j be the number of bytes actually read. Place the read bytes into the allocated block starting from index 1 so as to not overwrite the 0xFF. (b) Using mpz_import(), convert the read bytes, including the prepended 0xFF into an mpz_t m. You will want to set the order parameter of mpz_import() to 1 for the most significant word first, 1 for the endian parameter, and 0 for the nails parameter. (c) Encrypt m with

ss _encrypt(), then write the encrypted number to outfile as a hex string followed by a trailing newline.

void ss _decrypt_file(FILE *infile, FILE *outfile, mpz_t n)

- Will initiate variables and set them to NULL
    - M, fnl_decy
- Will initiate k which is a variable that stores the value of sizeinbase (n, 2)-1 /8 as a size_t type - Will use malloc to dynamically allocate arrays. Using uint8_t
- As mentioned in the a ss ignment doc will do a[0] = 0xFF
- And set a new size_t variable s as explained in the a ss ignment doc (its to print a certain number of bits)
- Initiate a while conduction that will run till it is not the end of a file with the parameter in file - Will use gmp_fscanf to read ile
    - Will call rsa_decrypt which will cal powmod
    - Will use mpz import to the order parameter of mpz_import() to 1 for the most significant wordfirst, 1 for the endian parameter, and 0 for the nails parameter
    - Will use fwrite to write to outfile
- Outside loop will use mpz_clears to clear all variables

void ss _decrypt(mpz_t m, mpz_t c, mpz_t d, mpz_t pq)

- I will call the pow mod function and pass in necessary values

<u>Key Generator</u>

This file is the main file with the main function in it. It can take the options of:

- Within main function will int opt and set it to 0
- Have a bool value called check for verbose
- If the user pa ss es a flag then the check will become true and it will print
    - - username,
    - first prime number with bits witch
    - second prime with bits then
    - modules of certain bit and and
    - privater exponent with certain bits
- Then will use uint 64 to initiate user to 256 as mentioned in the a ss ignment
- Will set uint 64 to iters to 50
- Will set int64 seed to time and pa ss NULL
- Then will create a pointer to pub file and private file
- Then I created pointers to variables that point to the ss pub and ss priv. - Will initiate a while loop with getopt
    - Will ninitate switch(opt) and define all the cases in the switch
        - Case b
            - In the case i will set pud_mod_n to atoi(optarg)
        - Case i
            - Set iters using atoi(optarg)
        - Case n
            - Set pointer public file atoi(optarg)

- Case d
    - Set pointer private file atoi(optarg)
- Case s
    - Set seed to atoi(optarg)
    - And then set randstad(seed)
- Case v
    - Change check value to true if this case is called so the verbose output is given
- Case h

```
fprintf(stderr, "SYNOPSIS\n");
        fprintf(stderr, "   Encrypts data using SS encryption.\n");
        fprintf(stderr, "   Encrypted data is decrypted by the decrypt
program.\n\n");
        fprintf(stderr, "USAGE\n");
        fprintf(stderr, "   ./encrypt [OPTIONS]\n\n");
        fprintf(stderr, "OPTIONS\n");
        fprintf(stderr, "   -h          Display program help and usage.\n");
        fprintf(stderr, "   -v          Display verbose program output.\n");
        fprintf(stderr, "   -i infile     Input file of data to encrypt (default:
stdin).\n");
        fprintf(
            stderr, "   -o outfile     Output file for encrypted data (default:
stdout).\n");
        fprintf(stderr, "   -n pbfile      Public key file (default:
ss.pub).\n");
```

    -
- For case ?
    - Will print same help me ss ages a above case

- Will open public and private file and check if they are empty, if value equal to NULL then would return file cannot be opened
- Now I will set the public mod n to mod variable
- Now I will do ss make pub and make priv
- Will set char pointer username and use getenv(“USERNAME”)
- Will ste using mpz_set and username, new_name with a base 62
- Then I did the rsa_sign and passed the parameters encry_s and new_name and username and pubfile
- Now I will call rsa_write pub and rsa_write prive to pa ss the fnl output to the public and private files respectively,
- Will have a check to see if vlag v for verbose was called then checker will be true which will print all the prints such as username ect.
- My fclose my pub and private files

Encrypt

- Within main function will int opt and set it to 0
- Have a bool value called check for verbose
- If the user pa ss es a flag then the check will become true and it will print
    - - username,
    - first prime number with bits witch
    - second prime with bits then
    - modules of certain bit and and
    - privater exponent with certain bits
- Then will use uint 64 to initiate user to 256 as mentioned in the a ss ignment
- Will set uint 64 to iters to 50
- Will set int64 seed to time and pa ss NULL
- Then will create a pointer to pub file and private file
- Then I created pointers to variables that point to the ss pub and ss priv. -

Will initiate a while loop with getopt

- Will ninitate switch(opt) and define all the cases in the switch

    This file is meant to act as the main function for the encryption program. This program will have the flags of:

    • -i : specifies the input file to encrypt (default: stdin).
    • -o : specifies the output file to encrypt (default: stdout).
    • -n : specifies the file containing the public key (default: ss .pub).
    • -v : enables verbose output.
    • -h : displays program synopsis and usage

    The function will use ss _encrpyt_file and use the mpz_t to encrypt the file. It will take in the username and the public key n,. The program will use get opt to set these flags an fopen and close to manipulate the in and out files from the other functions.

- I will open the public_key using fopen
- Will have an if that will check if any of the files that are going to be opened are NULL
    - Will use printf to print out an error message if so.
- WIll read using the ss_read_pub and pass in n and user_name and private temp file.
- Then I will have check if verbose is true or false
    - If true will print the bits
- Else will ignore
- Then will call the ss_encrypt_file function to carry out the encryption process.
- will close file using fclose()

Decryptor

Within main function will int opt and set it to 0

- Have a bool value called check for verbose

- If the user pa ss es a flag then the check will become true and it will print
    - - username,
    - first prime number with bits witch
    - second prime with bits then
    - modules of certain bit and and
    - privater exponent with certain bits
- Then will use uint 64 to initiate user to 256 as mentioned in the a ss ignment
- Will set uint 64 to iters to 50
- Will set int64 seed to time and pa ss NULL
- Then will create a pointer to pub file and private file
- Then I created pointers to variables that point to the ss pub and ss priv. -

Will initiate a while loop with getopt

- Will ninitate switch(opt) and define all the cases in the switch

This file is meant to act as the main function for the encryption program. This program will have the flags of:

This file will be the decrpyteor file. It will take the flash of:

• -i : specifies the input file to decrypt (default: stdin).
• -o : specifies the output file to decrypt (default: stdout).
• -n : specifies the file containing the private key (default: ss .priv).
• -v : enables verbose output.
• -h : displays program synopsis and usage.

This file will use the private modulus pq and the private key d to decrypt the message that came in through the output file of the other file.

- I will open the public_key using fopen
- Will have an if that will check if any of the files that are going to be opened are NULL
    - Will use printf to print out an error message if so.
- WIll read using the ss_read_priv and pass in n and user_name and private temp file.
- Then I will have check if verbose is true or false
    - If true will print the bits
- Else will ignore
- Then will call the ss_decrypt_file function to carry out the decryption process.
- will close file using fclose()

Randstatate

Will set gmp_ranstate_t and state

void randstate_init(uint64_t seed)

- This function will use a srandom seed and generate the state seed.

Void rnadtstate_clear(void)

- Will use gmp_rnadclear and pass in state.