Assignment 5 Design Document

Alex Yeh

February 24, 2023

1 Description of Assignment

This assignment is to create three programs (keygen, encrypt, and decrypt), two libraries (numtheory and ss), and a random state module (randstate). In order to complete all of these programs/libraries, I needed to learn how to use the GNU multiple precision arithmetic library (GMP).

2 Files to be included in directory "asgn5":

1. decrypt.c
    - This C file contains the implementation and main() function for the decrypt program.
2. encrypt.c
    - This C file contains the implementation and main() function for the encrypt program.
3. keygen.c
    - This C file contains the implementation and main() function for the keygen program.
4. numtheory.c
    - This C file contains the implementations of the number theory functions.
5. numtheory.h
    - This header file specifies the interface for the number theory functions.
6. randstate.c

- This C file contains the implementation of the random state interface for the SS library and number theory functions.

7. randstate.h
    - This header file specifies the interface for initializing and clearing the random state.
8. ss.c
    - This C file contains the implementation of the SS library.
9. ss.h
    - This header file specifies the interface for the SS library.
10. Makefile
    - This file directs the compilation process of keygen.c, encrypt.c, and decrypt.c.
11. README.md
    - This file is in Markdown format and describes how to use my program and Makefile. It also lists and explains the different command-line options that my programs accept.
12. DESIGN.pdf
    - This file is a PDF version of this design document for assignment 5. It describes my design and design process for my programs with pseudocode and images.
13. WRITEUP.pdf
    - This file is a PDF version of my writeup for assignment 5. It includes everything that I learned in this assignment. It also includes the applications of of public-private cryptography, how it influences the world today, and one way that I personally take advantage of public-private cryptography on a day-to-day basis.

3 Pseudocode

3.1 randstate.c

Initialize gmp_randstate_t state
randstate_init(uint64_t seed)
    Initialize the global random state with a Mersenne Twister algorithm using gmp_randinit_mt().
    Seed the random state with the seed using gmp_randseed_ui()
    Get random seed using srandom()

randstate_clear(void)
Clear and free all memory used by the random state using gmp_randclear().

3.2 numtheory.c

void gcd(mpz_t g, mpz_t a, mpz_t b)
Define mpz_t variables “t”, “temp_a”, and “temp_b”
Initialize and set temp_a to a
Initialize and set temp_b to b
Initialize t
While temp_b is not greater than 0
Set t to temp_b
Set b equal to a mod(b)
Set temp_a to t
Set output g to temp_a
Clear mpz_t variables

```
GCD(a, b)
1  while b ≠ 0
2      t ← b
3      b ← a mod b
4      a ← t
5  return a
```

Pseudocode for gcd from the assignment 5 pdf which I followed to write my gcd function

void mod_inverse(mpz_t o, mpz_t a, mpz_t n)
Define mpz_variables “r”, “newr”, “t”, “newt”, “q”, “temp” and “temp2”
Initialize and set r to n
Initialize and set newr to a
Initialize and set t to 0
Initilaize and set newt to 1
Initialize remaining mpz_t variables
While newr is not equal to 0
Set q equal to r divided by newr
Set temp to newr
Set temp2 equal to newr multiplied by q
Set newr equal to r - temp2
Set r to temp
Set temp equal to newt
Set temp2 equal to newt multiplied by q
Set newt equal to t subtracted by temp2
Set t equal to temp
If r is greater than 1
Set output o to 0
Return
If t is less than 0
Set t equal to t + n
Set output o to t
Clear mpz_t variables

```
MOD-INVERSE(a, n)
1   (r, r′) ← (n, a)
2   (t, t′) ← (0, 1)
3   while r′ ≠ 0
4       q ← ⌊r/r′⌋
5       (r, r′) ← (r′, r − q × r′)
6       (t, t′) ← (t′, t − q × t′)
7   if r > 1
8       return no inverse
9   if t < 0
10      t ← t + n
11  return t
```

Pseudocode for mod_inverse from the assignment 5 pdf which I followed to write my mod_inverse function

void pow_mod(mpz_t o, mpz_t a, mpz_t d, mpz_t n)

Define mpz_t variables “v”, “p”, “temp_d”, and “temp_p”
Initialize and set v to 1
Initialize and set p to a
Initialize and set temp_d to d
Initialize temp_p
While temp_d is greater than 0
If temp_d is odd
Set v equal to v multiplied by q
Set v equal to v mod n
Set temp_p to p
Set p equal to temp_p multiplied by temp_p
Set p equal to p mod n
Set temp_d equal to temp_d divided by 2
Set output o to v
Clear mpz_t variables

```
POWER-MOD(a, d, n)
1  v ← 1
2  p ← a
3  while d > 0
4       if ODD(d)
5            v ← (v × p) mod n
6       p ← (p × p) mod n
7       d ← ⌊d/2⌋
8  return v
```

Pseudocode for pow_mod from the assignment 5 pdf which I followed to write my pow_mod function

static void find_s_and_r(mpz_t s, mpz_t r, mpz_t n)

This function is used to calculate s and r to write the first line of the pseudocode from the assignment 5 pdf down below.

Define mpz_t variables “temp_n” and “temp_n2”
Initialize mpz_t variables
Set temp_n to n
Set temp_n2 to temp_n mod 2
While temp_n is not equal to 0
Set s equal to s plus 1
Set temp_n equal to temp_n divided by 2
Set temp_n equal to temp_n mod 2
Set r to temp_n
Clear mpz_t variables

bool is_prime(mpz_t n, uint64_t iters)

If n is equal to 0
Return 0
If n is equal to 1
Return 0
If n is equal to 2
Return 1
If n is equal to 3
Return 1
Define mpz_t variables “n_1”, “r”, “s”, “y”, “j”, “a”, “two”, “o”, “value”, and “s_1”
Initialize mpz_t variables
Set value equal to 2
Set n_1 equal to n-1
Set s equal to 0
Set r equal to 1
Call find_s_and_r function
Set s_1 to s - 1
If y is not equal to 1 and y is not equal to n-1
Set j to 1
While j is less than s-1 and y is not equal to n-1
Set o equal to yv̂alue mod n
Set y to o
If y is equal to 1
Clear mpz_t variables
Return 0
Set j equal to j plus 1
If y is not equal to n-1
Clear mpz_t variables
Return 0
Clear mpz_t variables
Return 1

MILLER-RABIN(n,k)

1 write $n-1=2^s r$ such that r is odd
2 **for** $i \leftarrow 1$ **to** k
3 choose random $a \in \{2,3,\ldots,n-2\}$
4 $y = \text{POWER-MOD}(a,r,n)$
5 **if** $y \neq 1$ and $y \neq n-1$
6 $j \leftarrow 1$
7 **while** $j \leq s-1$ and $y \neq n-1$
8 $y \leftarrow \text{POWER-MOD}(y,2,n)$
9 **if** $y == 1$
10 **return** FALSE
11 $j \leftarrow j+1$
12 **if** $y \neq n-1$
13 **return** FALSE
14 **return** TRUE

Pseudocode for is_prime from the assignment 5 pdf which I followed to write my is_prime function

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

do
Generate new prime number p
Set p to bits number of bits
while p is not prime

3.3 ss.c

void ss_make_pub(mpz_t p, mpz_t q, mpz_t n, uint64_t nbits, uint64_t iters)

Define mpz_t variables “q_1”, “q_1_mod_p”, “p_1”, “p_1_mod_q”, and “two_p”
Initialize mpz_t variables
Do
Define end_range as (2*nbits/5)
Set number of pbits to random number in range from nbits/5 to end_range
Set number of qbits to nbits - 2*pbits
Create prime p
Create prime q
Set q_1 equal to q-1
Set q_1_mod_p equal to q_1 mod p
Set p_1 equal to p-1
Set p_1_mod_q equal to p_1 mod q
While q_1_mod_p is equal to 0 or p_1_mod_q is equal to 0
Set two_p equal to p multiplied by p
Set n equal to two_p multiplied by q
Clear mpz_t variables

void ss_make_priv(mpz_t d, mpz_t pq, mpz_t p, mpz_t q)
Define mpz_t variables “p_1”, “q_1”, “pq_1”, “temp”, “lambda_pq”, “two_p”, and “n”
Initialize mpz_t variables
Set p_1 equal to p-1
Set q_1 equal to q-1
Set pq_1 equal to (p-1)(q-1)
Set pq equal to p multiplied by q
Set temp equal to the gcd of p_1 and q_1
Set lambda_pq equal to pq_1 divided by temp
Set two_p equal to p multiplied by p
Set n equal to two_p multiplied by q
Calculate the inverse of n mod lambda_pq
Clear mpz_t variables

void ss_write_pub(mpz_t n, char username[], FILE *pbfile)
Write public key to pbfile in the format n(hexstring), username each with a trailing newline using gmp_fprintf

void ss_write_priv(mpz_t pq, mpz_t d, FILE *pvfile)
Write private key to pvfile in the format pq(hexstring), d(hexstring) each with a trailing newline using gmp_fprintf

void ss_read_pub(mpz_t n, char username[], FILE *pbfile)
Read public key from pbfile in the format n(hexstring), username each with a trailing newline using gmp_fscanf

void ss_read_priv(mpz_t pq, mpz_t d, FILE *pvfile)

Read private key from pvfile in the format pq(hexstring), d(hexstring) each with a trailing newline using gmp_fscanf

void ss_encrypt(mpz_t c, mpz_t m, mpz_t n)

Set c equal to m^n mod n

void ss_encrypt_file(FILE *infile, FILE *outfile, mpz_t n)

Define mpz_t variables “m”, “c”, and “sqrt_n”
Initialize mpz_t variables
Set sqrt_n equal to the square root of n
Set k equal to (logbase2(sqrt_n)-1)/8
Allocate memory for kbytes
Set the zeroth byte to 0xFF
While not at end of file
    Set j to number of bytes read (read k-1 bytes)
    If j equals 0
        Break
    Import kbytes into m
    Encrypt m using ss_encrypt function
    Write m to outfile using gmp_fprintf
Clear mpz_t variables
Free kbytes

void ss_decrypt(mpz_t m, mpz_t c, mpz_t d, mpz_t pq)

Set m equal to c^d mod pq

void ss_decrypt_file(FILE *infile, FILE *outfile, mpz_t d, mpz_t pq)

Set k equal to (logbase2(pq)-1)/8
Allocate memory for kbytes
Define mpz_t variables “m” and “c”
Initialize mpz_t variables
Initialize size_t variable “j”
While not at end of file
    Read c from infile using gmp_fscanf and set to int variable x
    If x equals 0
        Break
    Decrypt c using ss_decrypt function back into m
    Convert m back to kbytes using mpz_export
    Write kbytes to outfile
Clear mpz_t variables

Free kbytes

3.4 keygen.c

define OPTIONS “b:i:n:d:s:vh”

Create static void function called “program_usage” to print out help message

int main(int argc, char **argv)
Set default number of bits to 256
Set default number of iterations to 50
Set default public key file name to “ss.pub”
Set default private key file name to “ss.priv”
Set default seconds since UNIX epoch
Set boolean variable “verbose” to false
Set int variable “opt” to 0
While loop while opt is not equal to -1
Switch statement for opt
Case ’b’:
Set number of bits user input using strtoul function
Break
Case ’i’:
Set number of iterations user input using strtoul function
Break
Case ’n’:
Set public key file name to user input
Break
Case ’d’:
Set private key file name to user input
Break
Case ’s’:
Set seed to user input using strtoul function
Break
Case ’v’:
Set verbose to true
Break
Case ’h’:
Call program_usage function
Return 0
Default:
Call program_usage function
Return 1

Set pbfile to fopen public key file name with “w”
If pbfile is NULL
    Call perror function using the public key file name
    Return 1
Set pvfile to fopen private key file name with “w”
If pvfile is NULL
    Call perror function using the private key file name
    Return 1
Set permissions of pvfile to 0600 using fchmod() and fileno() functions
Initialize random state
Define mpz_t variables “p”, “q”, “n”, “d”, and “pq”
Initialize mpz_t variables
Make the public key using the ss_make_pub function
Make the private key using the ss_make_priv function
Get the current user’s name as a string using the getenv function
Write the public key to the public key file using the ss_write_pub function
Write the private key to the private key file using the ss_write_priv function
If verbose is true
    Print username
    Prine first large prime p
    Print second large prime q
    Print public key n
    Print private modulus pq
    Print private exponent d
Close public key file
Close private key file
Clear random state using gmp_randclear function
Clear mpz_t variables
Return 0

3.5 encrypt.c

Define OPTIONS “i:o:n:vh”

Create static void function called “program_usage” to print out help message

int main(int argc, char **argv)
    Define mpz_t variable “n”
    Initialize mpz_t variable
    Set username to 256 characters
    Set default input file to encrypt to stdin
    Set default output file to encrypt to stdout
    Set default public key file name to “ss.pub”

Set boolean variable “verbose” to false
Set int variable “opt” to 0
While loop while opt is not equal to -1
    Switch statement for opt
        Case ’i’:
            Set input file to user input using fopen
            Break
        Case ’o’:
            Set output file to user input using fopen
            Break
        Case ’n’:
            Set public key file name to user input using optarg
            Break
        Case ’v’:
            Set verbose to true
            Break
        Case ’h’:
            Call program_usage function
            Return 0
        Default:
            Call program_usage function
            Return 1
Set infile to fopen input file with “r”
If public key file is NULL
    Call perror function with the public key file name
    Return 1
If input file is NULL
    Call perror function with input NULL
    Return 1
Read public key from public key file using ss_read_pub function
If verbose is true
    Print username
    Print public key n
Encrypt input file using ss_encrypt_file function
Close input file
Close output file
Clear mpz_t variables
Return 0

3.6 decrypt.c

Define OPTIONS “i:o:n:vh”

Create static void function called “program_usage” to print out help message

int main(int argc, char **argv)

Define mpz_t variables “pq” and “d”
Initialize mpz_t variables
Set default input file to decrypt to stdin
Set default output file to decrypt stdout
Set default public key file name to “ss.priv”
Set boolean variable “verbose” to false
Set int variable “opt” to 0
While loop while opt is not equal to -1
Switch statement for opt
Case ’i’:
Set input file to user input using fopen
Break
Case ’o’:
Set output file to user input using fopen
Break
Case ’n’:
Set private key file name to user input using optarg
Break
Case ’v’:
Set verbose to true
Break
Case ’h’:
Call program_usage function
Return 0
Default:
Call program_usage function
Return 1
Set infile to fopen input file with “r”
If private key file is NULL
Call perror function with the private key file name
Return 1
If infile is NULL
Call perror function with input NULL
Return 1
Read private key from private key file using ss_read_priv function
If verbose is true
Print the private modulus pq
Print private key d
Decrypt input file using ss_decrypt_file function

Close input file
Close output file
Clear mpz_t variables
Return 0

3.7 Makefile

Set CC equal to “clang”
Set CFLAGS equal to “-Wall -Werror -Wextra -Wpedantic $(shell pkg-config –cflags gmp)”
Set LFLAGS equal to “$(shell pkg-config –libs gmp)”
Set “all” to keygen encrypt and decrypt
Set “keygen” dependencies on object files keygen.o, randstate.o, numtheory.o, and ss.o
Set “encrypt” dependencies on object files encrypt.o, randstate.o, numtheory.o, and ss.o
Set “decrypt” dependencies on object files decrypt.o, randstate.o, numtheory.o, and ss.o
Set default rule for creating .o from .c files
Set rule for make clean
Set rule for make format