Assignment 5 Design

Brian Quach

February 16, 2022

1 Introduction

The program is broken into three different executables which ultimately allow for the encryption and decryption of messages. The program implements the idea of public and private key pairs in order to generate encrypted messages. Anyone can encrypt using an available public key but only those with a private key can decrypt. In this case, implementation relies on the concepts of Euclid's theory. Keygen essentially creates two large primes to generate a modulus n in which an exponent e is derived such that d × e = 1 (mod n) which allows for $m^{ed} = m$. Therefore, $c = m^e$ will encrypt the message and $m = c^d$ will decrypt.

2 Necessary Files

1. decrypt.c: Contains main() for decryption of a encrypted file.
2. encrypt.c: Contains main() for encryption of an input file
3. keygen.c: Contains implementation for creation of public and private key files.
4. numtheory.c: Conatains the implementation of numerical theory functions such as pow_mod() and makes use of theories such as euclid's and Miller-Rabin.
5. numtheory.h: Contains declarations for numtheory.c functions as well as inclusion of necessary libraries.
6. randstate.c: implements functions for instantiating a random state using a mersenne twister as well as a clear for the state.
7. randstate.h: Contains the interface for randstate.c.
8. rsa.c: implements functions which perform the bulk of the programs such as encrypt, decrypt, and the creation of a public and private key.
9. rsa.h: Interface for rsa.c
10. Makefile: Contains build process for all programs which includes compilation using pkg-config. Make clean will remove any .o files and executables.
11. README.md: Explains how to run and build the program as well as any errors found by scan-build or valgrind.

12. DESIGN.pdf: Explains design process with supporting pseudo code as well as error handling and credit to third parties.

3 randstate

Randstate sets up the random number generator from the gmp library. It contains an initialize function which sets the seed and mersenne twister algorithm for generating random numbers. It also has a clearing function which removes the state array. Both of these functions are done using functions provided by the gmp library.

4 gmp

gmp is essentially a library for dealing with larger numbers where numbers are stored in allocated arrays. This means that variables such as mpz_t must be both initialized and cleared after use. Gmp also provides random number generating functions which utilize a state variable in order to create pseudo random numbers.

5 Numtheory.c

Number theory implements the various numerical theory functions necessary in computing rsa keys and decryption/decryption. One of the main theories used is Euclid's where powmod, gcd, and modular inverse are derived from. The miller-rabin probabilistic primality test is another mathematical theory that is used in the production of rsa keys. The pseudo code for the following theories are all largely documented within the asgn5 pdf file with some key differences when translating to c. The biggest difference being the usage of gmp in order to obtain arbitrary precision integers since doing so requires initialization, clearing, performing only one operation at a time, and call by reference. Another changing factor is the fact that c does not support parallel assignment meaning a temp variable must be instantiated before calculations begin. For example, compared to the powmod pseudo code in the asgn5.pdf the actual c code may look closer to

```
void powmod(out, base, exp, mod) {
    mpz initialize q, v
    mpz set v = 1
    mpz set q = base
    while exp > 0 {
        if exp is odd {
            v = v * p
            v = v % mod
        }
        p = p * p
        p = p % mod
    }
    out = v
    mpz clear q, v
}
```

An important note is that any function that uses a numtheory function must plan for the chance that the passed values will be altered. This is due to the fact that gmp values are passed by reference and so any loop or reuse of parameters must be substituted with a temp/dummy variable. Another important note is to remember to clear all mpz_t variables made inside the function before every possible return case.

6 RSA library

The rsa library defines many of the basic functions needed in order to generate keys, encrypt files, and decrypt files.

```
rsa_make_pub(p, q, n, e, nbits, iters) {
    bitsp = random number range (nbits/4 ... 3nbits/4) using srandom
    p = make_prime using bitsp
    q = make_prime using nbits - bitsp
    Set p and q to a prime number using make_prime()
    compute lambda(n)
    while (gcd(of lambda != 1) {
        random number size nbits using mpz_urandomb()
        gcd = gcd(random number, lambda(n))
    }
    e = random number
}

void rsa_write_pub(n, e, s, username[], pbfile) {
    print n e s username[] to pbfile all separated by newlines also as hex strings
}

void rsa_make_priv(d, e, p, q) {
    d = mod_inverse of (e, lambda(p - 1, q - 1))
}

void rsa_write priv(
    the same thing as write_pub but for n and d
}

void rsa_read_priv(
    read in n and d in hexstring each on its own line and store them in the given
    parameter variables
}

void rsa_encrypt(c, m, e, n) {
    c = powmod(m) using e as the exponent and n as the modulus
}
```

```
void rsa_encrypt_file(infile, outfile, n, e) {
    calculate block size k
    allocate an array of size k with each space being a byte
    set the first index of the array to 0xFF for padding
    while (not end of file) {
        j = bytes <= k - 1
        place read bytes into the allocated array starting from index 1
        import the allocated array into an mpz_t
        encrypt the mpz
        print the encrypted mpz to an outfile as a hexstring
    }
}

void rsa_decrypt(m, c, d, n) {
    powmod using c as the base, d as the exponent, and n as the modulus
}

void rsa_decrypt_file(infile ,outfile, n, d) {
    get block size k (should be the same as encrypt_file's k)
    allocate array of size k with 1 byte each index
    while (not end of infile) {
        read in hexstring from infile
        decrypt the hex string using rsa_decrypt
        export the decrypted mpz into the allocated array
        write the bytes out to outfile starting while ignoring the 0th index (padding)
    }
}

void sign(s, m, d, n) {
    pow mod the message where exponent = d and mod = n
    the message will be the username of the caller from environment variables
}

bool verify(m, s, e, n) {
    pow mod s where exponent = e and mod = n
    if pow_mod s is equal to m then the verification is true
    otherwise it is false
}
```

7 Keygen

Keygen just creates a key using the functions from the rsa library. Using rsa_make_pub and then rsa_write_pub in order to write out the public key <e, n> to some given file. The same is then done for private keys. Noteworthy things being to always check if the file pointer is NULL and the macros for using fileno to set the private key file permissions.

8 encrypt and decrypt

Both main programs are simplified given that the bulk of the work is done in the rsa library. Encrypt will begin by reading variables from a get opt loop. The values of e, n, and s will then be read from a public key file. The signature will then be verified in regards to the username and return false if they are not equal. The file is then encrypted using rsa_encrypt_file and then variables cleared. Decrypt is an even simpler program where it performs like encrypt except it does not perform any verification.

9 Errors

- If the username exceeds 256 characters then the program will cease to function since it is stored in a 256 sized array when being read.
- If a file cannot be opened then a error message is printed and the program is exited.

10 Credit

- The pseudo code given in the asgn5 pdf was used for many of the numerical theory functions.
- The gcd code in the cryptography lecture was used as a pseudo code reference.
- Used a 256 size array to store the username in encrypt from awano on the class discord
- The idea of using fread came from Eugene's lab section.