CSE 13S Winter Quarter 2022
Assignment 5: Public Key Cryptography

Description of the Program:

The goal of this assignment is to familiarize ourselves with RSA keys and create programs to create, encrypt, and decrypt a public key.

Helper functions randstate, numtheory, and rsa are used to create functions to help us implement keygen, encrypt and decrypt.

Keygen is used to create a public and private key. Encrypt is used to encrypt an input. Decrypt is used to decrypt an input.

Files to be included in the “asgn1” directory

- decrypt.c
- encrypt.c
- keygen.c
- numtheory.c
- numtheory.h
- randstate.c
- randstate.h
- rsa.c
- rsa.h

Layout:

- randstate.c
    - initialize the seeds and state for mersenne twister
    - we use
        - gmp_randinit_mt and gmp_randseed_ui
        - gmp_randclear
- numtheory.c
    - create program to mimic mpz library to perform operations like gcd, power-mod, etc.
- rsa.c
    - creates the bulk of what we use in our keygen, encrypt, and decrypt main programs
- keygen.c
    - computes public and private keys
- encrypt.c
    - encrypts input
- decrypt.c
    - decrypts input

Pseudocode:

```
- randstate_init
    - initialize random for mpz
- randstate_clear
    - clear randstate
- gcd
    - compute gcd of two numbers
    - pseudocode provided
- power-mod
    - compute exponent and mod by mod value
    - pseudocode provided
- Miller-rabin
    - primality tester
    - pseudocode provided
- make_prime
    - do:
        - create random number 2^bits - 1 of length
        - if random number is even subtract 1 to make odd
    - while (random number is prime)
- mod_inverse
    - mod inverse of two numbers
    - pseudocode provided
- rsa_make_pub
    - p = make_prime()
    - q = make_prime()
    - n = p * q
    - calculate lambda n
    - while e and p lambda n are co prime
        - e = mpz_urandomb()
- rsa_write_pub
    - print out n, e, s, username
- rsa_read_pub
    - read n, e, s, username
- rsa_make_priv
    - calculate lambda n
    - perform mod_inverse(d, e, lambda n)
- rsa_write_priv
    - print out n, d
- rsa_read_priv
    - read n, d
- rsa_encrypt
    - c = power mod of m, e, n
- rsa_decrypt
```

    - m = power mod of c, d, n
- rsa_sign
    - s = power mod of m, d, n
- rsa_encrypt_file
    - k = (log_2(n) - 1) / 8
    - read from infile
        - import to m
        - c = encrypt m, e, n
- rsa_decrypt_file
    - k = (log_2(n) - 1) / 8
    - scan from infile
        - m = decrypt c, d, n
        - export to array
        - print out block size k of array
- rsa_veryify
    - if t = powermod s, e, n
    - if t == m
        - return true
    - return false

- keygen.c
    - get user input for getopt
    - make public and private keys
    - sign using rsa_sign
    - print out username, p, q, n, e, d
- encrypt.c
    - get user input for getopt
    - read public from rsa.pub
    - verify using rsa_verify
    - encrypt the file
    - print out s, n, e
- decrypt.c
    - get user input for getopt
    - read private from rsa.priv
    - decrypt the file
    - print out n, d

Notes on Pseudocode:

- You want to initialize and set new variables as inputs in the numtheory and rsa functions as not to change variables you don’t want to change
- make sure to end mpz_inits and mpz_clears with NULL
- make sure to close all open files
- make sure to free all allocated memory