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

## Description of the Program:

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

## 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

## Pseudocode:

```
- power-mod
    - v = 1
    - p = a
    - while d > 0:
        - if d % 2 == 1:
            - v = (v * p) % n
        - p = (p * p) % n
        - d /= 2
    - return v
- Miller-rabin
    - pseudocode provided
- make_prime
    - prime = false
    - RAND_MAX
    - PRIMES = [2, 3, 5, 7, 11, … 1987]
    - while prime == false:
        - set seed to time
        - rand = rand(2 ** bits, RAND_MAX)
        - if rand % 2 == 0:
            - rand += 1
        - for prime in PRIMES
            - if rand % prime == 0:
                - prime = false
            - else:
                -   if miller_rabin(rand, iters) == true:
                    -   prime = true
                -   else:
                    -   prime = false
-   gcd
    -   pseudocode provided
-   mod_inverse
    -   pseudocode provided
-   rsa_make_pub
    -   p = make_prime()
    -   q = make_prime()
    -   lcm = (p * q) / (gcd(p, q)
    -   exponent = 0:
    -   while 0 = 0:
        -   rand = mpz_urandomb()
        -   if gcd(rand, lcm) == lcm:
            -   exponent = rand
            -   break
    -   return exponent
-   rsa_write_pub
    -   FILE *file;
    -   file = fopen(pbfile, “w”)
    -   for i in range([n, e, s]):
        -   fprintf(“%hex\n”, i)
    -   fprintf(username)
    -   fclose(file)
-   rsa_read_pub
    -   FILE *file
    -   file = fopen(pbfile, “r”)
    -   n  = fscanf(%d)
    -   … for the rest
-   rsa_make_priv
    -   lcm = (p - 1)  * (q - 1) / gcd(p - 1, q - 1)
    -   lcm_2 = p * q / gcd(p, q)
    -   d = lcm * lcm_ 2 * e
-   rsa_encrypt
    -   c = m ** e % n
-   rsa_decrypt
    -   m = c ** d % n
-   rsa_decrypt_file
    -   k = (log_2(n) - 1) / 8
    -   calloc(k * sizeof(uint8_t)
    -   FILE *in
    -   in = fopen(infile, “r”)
```

- FILE *out = fopen(outfile, “w”)
- fprintf(mpzexport(fscanf))
- fclose(out)
- fclose(in)
- rsa_sign rsa_verify both follow the same format
- m ** d % n