Zion Kang
zkang5@ucsc.edu
8 November 2022

CSE13S Fall 2022
Assignment 5: Public Key Cryptography
Writeup

Testing of Code:

- Randstate:
  - For generating random numbers, both in standard c library math and mpz vars, I included a main() in randstate.c and compiled it with clang $(pkg-config –cflags gmp) $(pkg-config –libs gmp) -Wall -Werror -Wextra -Wpedantic -o randstate randstate.c.
  - The tests in the main() included gmp_rand and random() functions that generated and stored random values into mpz and standard math vars and printed them out.
- Numtheory:
  - For testing math logic in my numtheory functions, I used the same methodology in compiling randstate.c and numtheory.c in testing my main()
  - The tests in main() included those seen below in the pictures. I ran my tests and manually checked against calculated results online.

```c
int main(void) {
        // testing pow_mod
        mpz_t o;
        mpz_t a;
        mpz_t d;
        mpz_t n;
        mpz_init(o);
        mpz_init_set_ui(a, 19);
        mpz_init_set_ui(d, 8);
        mpz_init_set_ui(n, 2);
        gmp_printf("pow_mod of base: %Zu, exponent: %Zu, modulus: %Zu = ", a, d, n);
        pow_mod(o, a, d, n);
        gmp_printf("%Zu\n", o);
        // testing is_prime
        mpz_t x;
        mpz_init_set_ui(x, 43);
        randstate_init(13371453);
        if (is_prime(x, 3)) {
                printf("%s\n", "is_prime = true");
        } else {
                printf("%s\n", "is_prime = false");
        }
        // testing make_prime
        mpz_t p;
        mpz_init(p);
        make_prime(p, 32, 4);
        gmp_printf("prime made = %Zu\n", p);
```

```c
// testing gcd
mpz_init_set_ui(a, 12385);
mpz_t b;
mpz_init_set_ui(b, 4395);
gcd(d, a, b);
gmp_printf("my gcd = %Zu\n", d);
mpz_gcd(d, a, b);
gmp_printf("lib gcd = %Zu\n", d);
// testing mod inverse
mpz_t w;
mpz_t y;
mpz_init_set_ui(w, 1234);
mpz_init_set_ui(y, 213);
mod_inverse(o, w, y);
gmp_printf("mod inv = %Zu\n", o);
```

- Rsa.c:
    - For testing the accuracy of my encrypt, mainly my rsa_encrypt_file function, I compared the results of my values with that of the distributed encrypt by running ./encryp-dist (-options).
    - As for testing my rsa_decrypt_file function, I took in the provided privkey and message from a CSE13s alum from the discord (with username of Atomic) and ran the following:

```
int main(void) {
        mpz_t n, e, s, d;
        mpz_inits(n, e, s, d, NULL);
        char username[] = "";
        FILE *pbfile;
        FILE *pvfile;
        pbfile = fopen("pubkey", "r");
        pvfile = fopen("privkey", "r");
        rsa_read_pub(n, e, s, username, pbfile);
        rsa_read_priv(n, d, pvfile);
        gmp_printf("e = %Zu\n", e);
        gmp_printf("d = %Zu\n", d);
        FILE *infile;
        FILE *outfile;
        infile = fopen("testin", "r");
        outfile = fopen("testout", "w");
        rsa_decrypt_file(infile, outfile, n, d);
}
```

    - Since this message was larger than that of the ones I was testing initially, it helped me realize when to instantiate the block.
    - After learning to pass in j as a pointer and the indexing of certain params in fwrite, I was able to get the desired output of chinese text (bing chilling XD).
- Keygen.c, Encrypt.c, Decrypt.c
    - To keep it simple, I tested these against their respective copies: ./keygen-dist (-options), ./encrypt-dist (-options), ./decrypt-dist (-options).