Assignment 6: Public Key Cryptography

By Derfel Terciano

Background

Public key cryptography is a security measure that can be used to help encrypt and decrypt files. This system uses pairs of keys that consists of a public and a private key. The most popular public key cryptography algorithm used today is the RSA algorithm which was created by Ronald (R)ivest, Adi (S)hamir, and Leonard (A)dleman.

In this assignment, we will be using the RSA algorithm in order to generate a public and a private key pair that will be used to help encrypt and decrypt our files. RSA uses the idea of factoring two very large numbers in order to encrypt a message. The algorithm makes use of creating a public and a private key. The public key is what is used to encrypt the file and this key is known to everyone which is why it is public. Once a message is encrypted with a public key, it can only be decrypted with a private key which is not available to everyone.

The public key uses the modulus of some n and the public exponent of e. The private key uses a private exponent of d which is to be kept very secret. In addition, the variables that are used to calculate d must be kept secret. That means p, q, and φ(n) must be kept secret as well. Fortunately, those variables can be discarded after.

Now, let's dive deeper into the specifics of this assignment. That way we can get a good understanding of how we will be encrypting files using public and private keys.

Number Theoretic Functions

In order for us to create a public or private key, we would need to first create some function that can handle the big parts of the calculations of the keys. The following functions that we will implement are what drive the RSA algorithm:

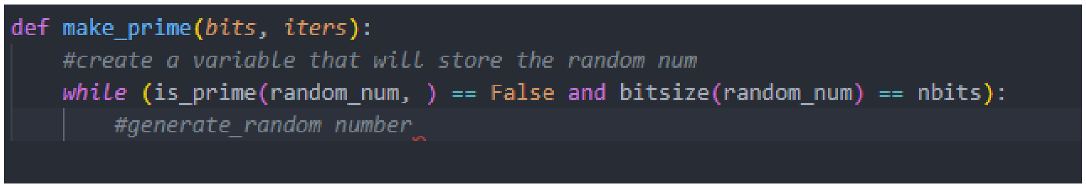
- Modular Exponentiation
- Miller-Rabin primality testing

- Modular Inverses

Modular Exponentiation

- void pow_mod(mpz_t out, mpz_t base, mpz_t exponent, mpz_t modulus)
    - This function uses a fast modular exponentiation technique that computes a base raised to the exponent mod n.
    - The pseudocode for this is not needed since it is provided in the assignment doc.

Miller-Rabin Primality Testing

- The main purpose of this function is to check whether or not a number is prime or not.
- void is_prime(mpz_t n, uint64_t iters)
    - This function uses the Miller-Rabin primality test to determine where or not *n* is prime or not. Iters determines how many iterations of the test we should use.
    - The pseudocode for this function is not needed since it already given to us in the assignment doc
- void make_prime(mpz_t p, uint64_t bits, uint64_t iters)
    - This function should generate a new prime number that is nbits long, and is prime using the is_prime() function.
    - Below is some python pseudocode I made:
    - 

```
def make_prime(bits, iters):
    #create a variable that will store the random num
    while (is_prime(random_num, ) == False and bitsize(random_num) == nbits):
        #generate_random number
```

Modular Inverses

- We will be needing to find the greatest common divisor in order to determine our public exponent. In this assignment, we will be using the Euclidean algorithm in order to find the greatest common divisor.
- void gcd(mpz_t d, mpz_t a, mpz_t b)
    - This function uses the Euclidean algorithm in order to find the gcd.

    - No pseudocode is needed here since it is already given to us in the assignment document
- void mod_inverse(mpz_t i, mpz_t a, mpz_t n)
    - This function computes the inverse of i of mod *n*. If we cannot find the inverse then we would return 0.
    - No pseudocode is needed for this since it is already provided in the assignment doc.