DESIGN - Assignment 6 - Public Key Cryptography

The objective of this assignment is to create a public and private key, and encode a file using RSA encryption. The reliability of this encryption is based on the difficulty of factoring the product of two very large prime numbers. The public key is known to the sender and the receiver, and only the private key can decrypt the message.

The first step is to create a public key:

Take two large prime numbers, p and q
Multiply them into the integer n
Take an exponent e that is coprime with (p - 1)(q - 1), or the totient of n
The public key is the pair <e, n>

With this exponent e and integer n, you can now apply the totient of n, and the decryption key d. The totient of a value is (value - 1). In the case of n, the totient is (p - 1)(q - 1). The formula for finding the decryption key d is as follows.

$e * d = 1 \text{ modulus } (\phi(n))$

This allows us, the encryptor, to obtain the decryptor d with relative ease, but keeps the value hidden to anyone who does not know what n is. The reason for this is behind the idea of the totient. It was already difficult to factor the product of two prime numbers, and it becomes nearly impossible when you're unaware of what the prime factors were before the totient. With this, the value of d remains unknown.

The first step in this process is to obtain two prime numbers. This number should be of a certain size, and so it's a matter of producing a prime number large enough to resist factorization, while still actually being prime. It would be terribly inefficient to test every prime number before it, especially on the scale that we are working on, so what comes into play now is the Miller-Rabin primality test. The primality test does not completely ensure that the number is prime, but with

enough iterations, it can essentially guarantee within an incredibly small margin of error that the number is prime:

For i in iterations:
    Choose a random number
    Test the primality of the prime number
    If it's not prime, return false
Return true

With the prime number p and q established, you can easily obtain the product p, the totient t, and the exponent e. The value of e is a random number, where the minimum number of bits was previously specified, and is coprime with the totient. The private exponent d is the inverse of e (mod n), and when the function is applied, can decrypt the encryption. The first step to this process is to encrypt the file:

Default the first value in the array
Find the block size
For each block in the file:
    Read in to the buffer
    Convert the buffer to an mpz number
    number = number ^ e (mod n)
    Write out to the outfile
Free the array and numbers

Once the file is encrypted, you can decrypt it, if you have access to the private key, which contains the public modulus n and the private exponent d. Since the encryption and decryption are inverse functions, the decryption follows the process of the encryption, in reverse order:

Find the block size
For each block in the file:
    Read in to a number

number = number ^ d (mod n)

Convert the number into a buffer

Write to the outfile from index 1

Free the array and numbers

Credit for the pseudocode provided and other helpful information goes to Professor Long and the staff of CSE-13S in the Fall 2021 quarter.