Assignment 5

James Contini

2022 October 26

General information

Citations

Assignment five pdf.

Description

This design document will be split into two pieces numtheory and usage. Numtheory will describe the number theory functions and usage will describe everything's implementation.

1 Miscellaneous

1.1 rand state init64()

The parameter for this function is parsed from the command line. It is a number which is set as the global seed for all deterministic random number generating functions.

1.2 rand clear()

This function clears the global random seed variable.

2 Numtheory

2.1 pow mod()

This function takes four parameters, an out, a base, an exponent and a modulus. Using the fact that any integer, can be represented as a polynomial, we can calculate the exponent out of a sum of base two polynomials. Similar to how decimal numbers are represented in binary. The function keeps calculating powers within *mod(n)* until we've looped all the way to the original exponent in *mod(n)* range.

2.2 is prime()

This function uses the Miller-Rabin primality test. For n iterations of checking the number against a witness, the test has a $1/4^{-n}$ chance of being wrong. Therefore the more iterations the more likely, the pseudo-prime will be truly prime.

2.3 make prime()

This functions takes three parameter generates. mpz out, n bits and iters iterations. Large random numbers of n bits are generated using the Mersenne Twister Algorithm and the seed provided by the rand state variable. The numbers are checked for primality using is prime() of *iters* iterations. If the large number is prime then it is placed in mpz out.

2.4 gcd()

This function calculates the greatest common divisor of two numbers using the Euclidean Algorithm. The greater number modulus the lesser number until the and place the modulus in the lesser number variable until it equals zero. Then return the modulus used that gave a remainder of zero.

3 encrypt

3.1 rsa make pub()

A public key n will be made by multiplying two large primes from make prime(). Then the public exponent will be made by picking a number coprime to n. Its easy to pick the number 65,536 because its prime and also just over four bytes long.

3.2 rsa write pub()

This writes the public keys to a .pub file. This function is used by rsa make pub().

3.3 rsa read pub()

This function parses the public key components to be used for encryption.

3.4 rsa encrypt file()

This parses chunks sized n from an infile. Each chunk is then mpz import()ed to be made into an mpz t. Then these are fed to rsa encrypt() to be encrypted into ciphertext which will then be appended to a outfile as hexstrings.

3.5 rsa encrypt()

One chunk of data can be encrypted at a time. The data is of size n. The encryption is performed by raising the data to the public exponent e and then taking the modulus by n. This will render c, the ciphertext which will be accessible in the form of a mpz t variable.

4 decrypt

4.1 rsa make priv()

This function calculates the private key using the Euler-Fermat theorem to come up with a private key d that when multiplied to e in the exponent will neutralize it and turn C back into M.

4.2 rsa write priv()

This function is similar to make public, it just writes the private key to a .priv file which will be used for decryption.

4.3 rsa read priv()

This function has permissions to read the .priv to acquire the private key which will be used for decrypting ciphertext.

4.4 rsa decrypt file()

This function reads from an infile where it feeds cipher text to rsa decrypt() one chunk at a time and then the original message is written to an outfile.

4.5 rsa decrypt()

Similar to rsa encrypt() this function assumes a chunk sized n has been fed to it. The function will then compute the original message M using the private key d to neutralize the public exponent e.