Tanya Gyanmote
1887664
tgyanmot@ucsc.edu

DESIGN DOC

Randstate.c

- Randstate_init
    - Create a random seed using the gmp_randstate_t state in randstate.h
    - Call gmp_randinit_mtwith the gmp_randstate_t state
    - Call gmp_randseed_ui with the gmp_randstate_t state and , the uint_64 seed
- Randstate_clear
    - Call gmp_randclear with gmp_randstate_t state
    - Clearing memory out of state

Numtheory.c

- Pow_mod
    - Creating temporary mpz values
    - Setting them to null
    - Initializing temporary V to 1
    - Initializing temporary P to a
    - Initializing temporary D to d
    - Have a while loop to check if temporary d is greater than 0
        - If the value of v is odd then set that value to v multiplied with v times p modulus n
        - After multiply p with p and set it to p
        - Set p to p multiplied with p modulus n
        - Set d to d divided by 2
    - Setting my o variable to my temporary v variable
    - Clearing all the values I created
- gcd
    - Creating temporary mpz values
    - Setting them to null
    - Initializing temporary a to a
    - Initializing temporary b to b
        - Having a whale loop to check if temporary b is not equal to 0
        - Create a temporary variable for t
        - Set the value of t to b
        - Set temporary b value to a modulus b
        - Set temporary a variable to t
    - Set temporary d variable to a

- make_prime
    - generates a new P prime numbe
    - P should be bits number of bits
    - the generated number should be tested using is_prime()with iters
- mod_inverse
    - Creating temporary mpz values
    - Setting them to null
    - Initializing temporary r to n
    - Initializing temporary r prime to a
    - Initializing temporary t to 0
    - Initializing temporary t prime to 1
    - While r prime isn't 0
        - Creating temporary mpz values
        - Setting them to null
        - Setting temporary q to r divided by r prime
        - Seting temporary r value to the r prime
        - Seting temporary r prime value to r minus q times r prime
        - Seting temporary t value to the t prime
        - Seting temporary t prime to t minus q times t prime
        - Clear all temporary variables
    - If the statement for r is greater than 0
        - Set temporary t to 0
    - If the statement for that t is greater than 0
        - Set temporary t to t plus n
    - Set out to t
    - Clear temporary variables

- Is_prime
    - Creating temporary mpz values
    - Setting them to null
    - Creating a odd R value
        - Have an equation for $n-1 = s^s r$
    - For loop starting at 1 until its iters
        - Using the random generator for the a value
        - Using power mod with inputs set to y
            - If y isn't 1 and n minus 1
                - Setting variable j to 1
                - While j isn't equal to n miinus 1 and it's less than s minus 1
                - Set power_mod of y,2,n all to y

            - If y is 1
                - Bool return false
            - Add one to the j value
        - If the y variable isn't equal to n minus 1
            - Return bool false
        - Return bool true

Decrypt.c

- Create options
- Using get opt
    - case i
        - input file to decrypt stdin
    - case o
        - decrypt with stdout
    - case n
        - the file with private key and rsa.priv
    - case v
        - verbose output prints:
            - public modulus n and private key
            - number of bits and values in decimal
    - case h
        - prints synopsis
- call the fopen
    - private key file
    - If it doesn't then print an error message plus return 0
- call the rsa_read_priv
- call the rsa_encrypt_file
- clear all the mpz variables

Encrypt.c

- Create options
- Using get opt
    - case i
        - input file to encrypt stdin
    - case o
        - encrypt with stdout
    - case n
        - the file with public key and rsa.pub
    - case v

        - verbose output prints:
            - public modulus n and private key
            - number of bits and values in decimal
    - case h
        - prints synopsis
- call the fopen
    - public key file
    - If it doesn't then print an error message plus return 0
- call the rsa_read_prub
- call the rsa_encrypt_file
- set the correct username to an mpz
- if the RSA verify the username is false
    - print the reported error and return 0
- lastly, clear all the mpz variables changes/used in this file

Keygen.c

- Create options
- Using get opt
    - case b
        - specify the minimum bits needed
    - case i
        - specify the number of Miller-Rabin iterations for testing primes
    - case n
        - the file with public key and rsa.pub
    - case d
        - the file with private key and rsa.priv
    - case s
        - for random seed
    - case v
        - verbose output prints:
        - username
        - signature
        - large prime, and the next large prime
        - the public modulus
        - public exponent
    - case h
        - prints synopsis
- Using fopen to open both public, private key files
    - If it doesnt work return 0 and error

- Making sure the file permissions are 0600
- Calling init function in randstad to set seed
- Using closer to close both public and private files
- Clear all mpz variables

Rsa.c

rsa_make_priv
- calling the mod inverse of inputs

rsa_make_prub
- calling make prime
- If the log is greater than nbits
    - P is the random number in range of (nbits/4,(3 times nbits) divided by 4
    - Q is the remaining
- Using random with iters to get a random value
- Go through the least common multiple with the greatest common denominator
    - The least common multiple
- For loop to call random numbers
    - Calling the mpz random function
    - Compute the gcd

rsa_write_priv
- using fopen
- print %x twice
- using fclose

rsa_write_pub
- using fopen
- print %x twice
- using fclose

rsa_read_priv
- using fopen
- using gmp scanf to read the hexstring
- using fclose

rsa_read_pub
- using fopen

- using gmp to scanf pbfile and username
- using fclose

rsa_encrypt
- ciphertext
- encrypting the message m
- using power mod with c,m,e,n

rsa_decrypt
- ciphertext with private key
- get message m
- encrypting the message m
- using power mod with c,m,d,n

rsa_sign
- calling power mod with s,m,d,n
- setting value of s to the value from power_mod

rsa_verify
- using power mod with t, s, e, n
- if value t and m are equal then true
- else false