Mahyar V.

11/23/2021

ASGN6 Design Doc

Purpose:

In this assignment, we are going to design a program that encrypts and decrypts messages. This is possible through a mixture of public key and symmetric key cryptography. Cryptography works using an algorithm that uses pairs of public (known to others) and private (known by the owner) keys. The same cryptographic keys are used for the encryption of plaintext and the decryption of ciphertext.

rsa.c: This is the program for handling the public keys. It can read them in from a file and write them to an outfile. It can also verify signatures and return true or false if they match. The file is compiled by make rsa.

Randstate.c: This file sets the random value for the state.

Numtheory: This program handles all the math used in the assignment. The GCD function computes the greatest common denominator. The isPrime returns a bool value corresponding to the primality of the function. makePrime will store a prime number in the given mpz value. The program is compiled using make Numtheory.

Keygen.c: This program generates the public key for the encryption algorithm. It prints all the command line outputs and it also grabs the user's signature from their device using getenv(). It can also print all the values if the user wants the verbose definition. The files are all written to an outfile. It is compiled by make Keygen.

Encrypt.c/Decrypt.c: This file prints the command line output and encrypts/decrypts the infile. The public/private key is printed to an outfile.

Pseudocode:

Randstate.c

randstate_init(uint64_t seed):	**randstate_clear(void):**
Simply call the two functions: gmp_randinit_mt() gmp_randseed_ui(seed)	call to gmp_randclear()

Numtheory.c

void gcd(mpz_t d, mpz_t a, mpz_t b):	**void mod_inverse(mpz_t i, mpz_t a, mpz_t n):**

// make sure to use gmp library
Use the comparison function to check:
while b is not equal to 0:
  Create and set mpz aa, bb, temp
  While bb is not 0:
  Set temp. to bb
  Set bb to aa mod bb
  Set a to temp.
  Set the variable d to aa
  Clear all variables
	Create r, inverse, t, tinvers, q, temp
	(r,rinv) is set to (n,a)
	(t,tinv) is set to (0,1)
	while rinv is not 0
	  q is set to mpz floor div. (r, ring)
	  Temp is set to r
	  (r,r inverse) is set to (r inverse , mpz mul(q, rinv)
	  Set rinv to temp - rinv
	  Set temp to t
	  (t,t inverse) is set to (t inverse, mpz mul(q, tinv))
	if r > 1, set i to 0 and clear variables
	if t < 0, t is set to t+n
	return t and clear variables

	Set i to t, clear variables

void pow_mod(o, a, d, n):
Create and set 1 to variable “v”
Create and set “a” to variable p
while d is greater than 0:
  If d is odd
  Create and set ((v * p) mod n) to variable “v”
  Set p to mpz mul (p, p)
  Set p to mpz mod(p, n)
  Set d to mpz floor div(d, 2)
Set o to v, clear all variables
	bool is_prime(mpz_t n, uint64_t iters):
	Make a algorithm that finds values of “s” and “r”, r being odd and letting n-1=r(2^s)
	Loop from regular integer i = 0 to iters
	  Set “a” to a random mpz urandomb number(state, n”
	  Add “a” by two to fix the range of random number
	  Pow mod(y, a, r, n)
	  If ‘y’ is not 1 and not (n-1)
	  Set ‘j’ to 1
	  While ‘j’ is less then (s-1) and ‘y’ is not (n-1)
	  Pow mod(y, y, 2, n)
	  If ‘y’ is 1, clear variables and return false
	  Increment j
	  If ‘y’ is not (n-1) clear variables and return false
	Clear variables and return true

void make_prime(mpz_t p, uint64_t bits, uint64_t iters): While not is_prime() or is size in base(p, 2) less than bits Generate a urandomb(p, state, size of bits +2)	

rsa_make_pub(p, q, n, e, nbits, iters) Pbits =((random number) mod (nbits/2))add(nbits/4) qbits = nbits - pbits Make prime for p and q Set n to mpz mul (p, q) Set pmin to p -1; qmin to q -1 Set phi to mpz mul(pmin, qmin) While true: Choose random number using urandomb gcd(result, random, phi) If result is 1 return false Set e to random Clear variables	**rsa_write_pub(mpz_t n, mpz_t e, mpz_t s, char username[], FILE *pbfile)** Use file gmp print to print n, e, s, username in the pbfile
rsa_read_pub(mpz_t n, mpz_t e, mpz_t s, char username[], FILE *pbfile) Use file gmp scan to print n, e, s, username in the pbfile	**rsa_make_priv(mpz_t d, mpz_t e, mpz_t p, mpz_t q)** Set pmin to p -1; qmin to q -1 Set phi to mpz mul(pmin, qmin) Set d to mod inverse (e, phi) Clear variables
rsa_write_priv(mpz_t n, mpz_t d, FILE *pvfile) Use file gmp scan to print n, d in the pvfile	**rsa_read_priv(mpz_t n, mpz_t d, FILE *pvfile)** Use file gmp scan to print n, d in the pvfile

rsa_encrypt(mpz_t c, mpz_t m, mpz_t e, mpz_t n) pow_mod(c, m, e, n)	**rsa_decrypt(mpz_t m, mpz_t c, mpz_t d, mpz_t n)** pow_mod(m, c, d, n)
rsa_sign(mpz_t s, mpz_t m, mpz_t d, mpz_t n) pow_mod(m, s, d, n)	**rsa_verify(mpz_t m, mpz_t s, mpz_t e, mpz_t n)** Set temp to m Pow mod (temp, s, e, n) Check if temp equals m and return true else false Clear variables before returning bool

Keygen:

- Make necessary flags, variables, files, file names, mpz values
- Prompt command line using getopt, switch-cases
- Print a helper guide for each possible command options
- Open the public and private key files using fopen(filename, “w”). Print a helpful error and exit the program in the event of failure.
- Use fchmod(fileno(file), ___) to make sure that the private key file permissions are set to 0600.
- Initialize random state
- Using rsa_make_pub/priv create the keys
- Use getenv to get username and convert using mpz set str(62)
- Compute the signature of the username using rsa_sign
- Write the public and private keys to their files using rsa_write_pub/priv
- Enable verbose to print out: username, signature, first two large prime numbers, modulus, exponent, private key ‘d’

- Clear mpz values, clear randomstate, close files

Encrypt:

- Make necessary flags, variables, files, file names, mpz values
- Prompt command line using getopt, switch-cases
- Print a helper guide for each possible command options
- Set the public file using fopen(public file name, “w”)
- Use rsa_read_pub to read public file
- Print verbose: username, signature, modulus, exponent
- Use getenv to get username and convert using mpz set str(62)
- Verify signature using rsa_verify; if fails, print error and exit
- Encrypt file using rsa_encrypt_file, close all files, clear variables

Decrypt:

- Make necessary flags, variables, files, file names, mpz values
- Prompt command line using getopt, switch-cases
- Print a helper guide for each possible command options
- Set the private file using fopen(private file name, “w”)
- Use rsa_read_priv to read private file
- Print verbose: modulus, private key “e”
- Decrypt file using rsa_decrypt_file, close all files, clear variables