Guillermo Alfaro
CSE13S Winter 2022

Public Key Cryptography

Description:

Create programs that will encrypt and decrypt files. It will utilize three programs, one that generates two keys, another that encrypts based on the generated public key, and one that decrypts from the private key generated.

Files:

1. Decrypt.c

   This contains the implementation and main() function for the decrypt program.

2. Encrypt.c

   This contains the implementation and main() function for the encrypt program.

3. Keygen.c

   This contains the implementation and main() function for the keygen program

4. Numtheory.c

   This contains the implementations of the number theory functions.

5. Numtheory.h

   This specifies the interface for the number theory functions.

6. Randstate.c

   This contains the implementation of the random state interface for the RSA library and number theory functions.

7. Randstad.h

   This specifies the interface for initializing and clearing the random state.

8. Rsa.c

   This contains the implementation of the RSA library.

9. Rsa.h

    This specifies the interface for the RSA library.

10. Makefile
11. README.md

Pseudocode:

Numtheory.c

```
    Gcd
        Create temp variables to not modify original given
        While b > 0
            t = b
            b = a % b
            a = t
        Set d = a
        Clear temp variables
    Mod_inverse
        Create temp and in-between variables to mimic parallel assignment
        r = n; r’ = a;
        t = 0; t’ = 1;
        While r’ != 0
            (
    Pow_mod
```

```
        V = 1
        While exponent > 0
            If off
                V = (v * p) % modulus
                P = (p * p) % modulus
                Exponent =/ 2
        Return v

    Is_prime
        Use miller-rabin test pseudo code page 11 on pdf
    Make_prime
        (pending)
Randstad.c
    Randstad_init
        Use mersenne twister algorithm, seed is random input
    Randstate_clear
        Clears all memory, used by gmp using gmp_randclear()
RSA.c
    Ras_make_pub
        Generate P and Q using make_prime()
        Compute LCM(p-1, q-1)
        Generate usable exponent e
            Use mpz_urandomb()
```

Use gcd of each random num

Stop loop when coprime is found of lambda(n)

Rsa_write_pub

- Write public key to pbfile
- Format as n\n, e\n, s\n, username\n
  - N & E are hex

Rsa_read_pub

- Read public key from pbfile
- Format as n\n, e\n, s\n, username\n
  - N & E are hex

Rsa_make_priv

- Creates RSA private key d given p, q, & e
- D = inverse e modulo lambda(n) = lcm(p - 1, q - 1)

Rsa_write_priv

- Write private key to pvfile n then d

Rsa_read_priv

- Reads private key like written in ^

Rsa_encrypy_file

- E(m) = c = m^e (mod n)

Rsa_encrypt_file

- encrpts infile to outfile by blocks less than n
- Blocks cannot be 0 or 1

Rsa_decrypt

```
        Reverse of encrypt
    Rsa_decypt_file
        Reverse of encrypt file
    Rsa_sign
        S(m) = s = $m^d$
    Rsa_verify
        Returns true if signature is verified
        Inverse of signing
Decrypt.c
    main()
        Initizliase mpz_t’s
        FILE’s
        Bools
        Int opt = 0;

        While (getopt(“i o n v h”) != -1)
            Case for each input ^
        If file open is null
            put error message and return 0
        If v
            Print n and size
            Print d and size
        If h
```

```
            Print options

        rsa_decrypt(infile, outfile…)

        Close infile, and outfile

        Clear mpz_t

Encrypt.c

    Int main()

        Initizlse mpz_t's

        Int opt = 0

        FILE's

        Bools

        While loop getopt != -1

            Cases for potential arguments, assigning to their variables

        If file pointers is NULL

            Puts error message

            Exits program

        If v

            Print username, signature and size, n and size, e and size

        If h

            Puts help screen

        Verify that signature and username from rsa.pub match

        rsa_encrypt_file()

        Close rsa.pub

        Clear mpz_t's
```

Keygen.c

```
    Initialize mpz_t’s
    Initialize all types for get opt switch
    While (getopt(“b i n d s v h”))
        Case for each arguments
    If b < 0
        Print error message
        Clear mpz_ts
        quit
    If h
        Print help screen
    If s is initial value (there was no -s called)
        Call ranstate with time(NULL)
    else
        Call ranstate with s
    Check file permissions for private key
        If not already locked to user, then lock it to only user using fchmod
    rsa_make_pub()
    rsa_make_priv()
    Get username with getenv and set it to mpz
        rsa_sign()
    Write public keys and private keys to their files
    If v
```

```
        Print username, s, p, q, n, e, d and all their sizes

    Close all files

    Clear randstate

    mpz_clear()

return
```