Mylo Lynch

Darrel Long

Cse 13s

2/5/23

# ENCRYPTION

## Deliverables:

1. decrypt.c: contains main() for the decrypt program.

```
Decrypts data using an SS encryption.\n   Encrypted data is "

            "decrypted by the decrypt program.\n\nUSAGE\n   ./decrypt "

            "[OPTIONS]\n\nOPTIONS\n  -h\t\t\tDisplay program help and usage.\n  "

            "-v\t\t\tDisplay verbose program output.\n  -i infile\t\tInput file of data to "

            "encrypt (default: stdin).\n  -o outfile\tOutput file for encrypted data "

            "(default: stdout).\n  -n pbfile\t\tPublic key file (default: ss.pub).\n"
```

`   Open privkey file for usage

Read pq and d

if verbose print the verbose output

default input is stdin and output is stdout

Call ss_decrypt_file

Clear and free`

2. encrypt.c: contains Main() for the encrypt program.

   Encrypts data using an SS encryption.\n   Encrypted data is "

   "decrypted by the decrypt program.\n\nUSAGE\n   ./encrypt "

   "[OPTIONS]\n\nOPTIONS\n  -h\t\t\tDisplay program help and usage.\n  "

   "-v\t\t\tDisplay verbose program output.\n  -i infile\t\tInput file of data to "

   "encrypt (default: stdin).\n  -o outfile\tOutput file for encrypted data "

   "(default: stdout).\n  -n pbfile\t\tPublic key file (default: ss.pub).\n"

   `open the public file and Read public key (n) and username

   if -v is used in the command line print the verbose output

   default input is stdin and output is stdout

   Call ss_encrypt_file

   Clear and free`

3. keygen.c:generates keys for the encryption and decryption algorithms

   "SYNOPSIS:\n   Generates an SS public/private key pair.\n\nUSAGE\n   ./keygen "

```
        "[OPTIONS]\n\nOPTIONS\n  -h\t\tDisplay program help and usage.\n
-v\t\tDisplay "
        "verbose program output.\n  -b bits\tMinimum bits needed for public key n "
        "(default: 256).\n  -i iterations\tMiller-Rabin iterations for testing primes "
        "(default: 50).\n  -n pbfile\tPublic key file (default: ss.pub).\n  -d "
        "pvfile\tPrivate key file (default: ss.priv).\n  -s seed\tRandom seed for "
        "testing.\n"
```

```
`get username
initialize the random state
make public key and add public key to the file
make private key and add the private key to the file
set the file permissions to 0600
clear initialized variables`
```

4. numtheory.c: This contains the number theory functions in code

***For this we turn math into code, there is given pseudo code in the assignment doc that will be used in combination with the gmp library to make the numtheory functions.***

***TURN THE PSEUDOCODE GIVEN INTO A FORMAT THAT THE ALLOWED MPZ FUNCTIONS WILL BE ABLE TO SUPPORT. BASICALLY ADD AN EXTRA PARAMETER FOR THE OUTPUT AND DON'T TREAT THESE LIKE INTEGERS, BUT AS SPECIAL VALUES***

5. numtheory.h:header for number theory functions.
6. randstate.c: random state interface for the SS library and number theory functions.

Initialize variable state then for randstate_init do gmp_randinit_mt(state); gmp_randseed_ui(state, seed);

Also make a function to clear the state variable

7. randstate.h:header for initializing and clearing the randomstate.
8. ss.c:SS library code, this is where the majority of the logic will be. Encrypt and Decrypt and Keygen will use functions from this library in order to encrypt and decrypt the given messages.

**void ss_make_pub(mpz_t p, mpz_t q, mpz_t n, uint64_t nbits, uint64_t iters)**

Will make a public key

`calculate bits needed for p

Calculate bits needed for q

Get n = p^2*q (n is public key)`

**void ss_write_pub(const mpz_t n, const char username[], FILE *pbfile)**

`print n to file as a hexstring then print username to file`

**void ss_read_pub(mpz_t n, char username[], FILE *pbfile)**

`read hexstring and set n to that value then read username and clean up extra line from it`

**void ss_make_priv(mpz_t d, mpz_t pq, const mpz_t p, const mpz_t q)**

`find lcm using (p-1)(q-1)/gcd

Get n mod inv lcm = d

Clear variables`

**Ss_encrypt and ss_decrypt** are just using the pow_mod function from numtheory

**ss_write_priv(const mpz_t pq, const mpz_t d, FILE *pvfile)**

`make into hex string

Print to file`

**void ss_read_priv(mpz_t pq, mpz_t d, FILE *pvfile)**

` get pq and turn it into mpz_t like every other variable this code works with

get d and turn it into mpz_t like every other variable this code works with`

**void ss_encrypt_file(FILE *infile, FILE *outfile, const mpz_t n)**

` find the size of the block to be able to allocate the proper amount of memory for the block

0xFF should be the value of the first block byte only able to read block_size - 1 max

if less, then the file is done and pad remaining space with 0

Change block to mpz_t

Encrypt and write output to outfile then free block`

**void ss_decrypt_file(FILE *infile, FILE *outfile, const mpz_t d, const mpz_t pq) {**

`find the size of the block to be able to allocate the proper amount of memory for the block

go through all lines within the infile

Scan hexstring and convert to mpz_t

initialize message and set value to decrypted crypt

convert m into bytes by using mpz_export and put them in the block

once converted into bytes we're able to finally write out our actual message

Clear and free`

9.ss.h: header for the SS library.

**Makefile**

```
CC = clang
CFLAGS = -Wall -Wextra -Werror -Wpedantic $(shell pkg-config --cflags gmp)
LFLAGS = $(shell pkg-config --libs gmp)
```

```
TARGETS = keygen encrypt decrypt
OBJS = randstate.o numtheory.o ss.o keygen.o encrypt.o decrypt.o

all: keygen encrypt decrypt

keygen: keygen.o numtheory.o ss.o randstate.o
	$(CC) $(CFLAGS) -o $@ $^ $(LFLAGS)

encrypt: encrypt.o ss.o numtheory.o randstate.o
	$(CC) $(CFLAGS) -o $@ $^ $(LFLAGS)

decrypt: decrypt.o ss.o numtheory.o randstate.o
	$(CC) $(CFLAGS) -o $@ $^ $(LFLAGS)

randstate.o: randstate.c
	$(CC) $(CFLAGS) -c $<

numtheory.o: numtheory.c
	$(CC) $(CFLAGS) -c $<

ss.o: ss.c
	$(CC) $(CFLAGS) -c $<

keygen.o: keygen.c
	$(CC) $(CFLAGS) -c $<

encrypt.o: encrypt.c
	$(CC) $(CFLAGS) -c $<

decrypt.o: decrypt.c
	$(CC) $(CFLAGS) -c $<

%.o: %.c
	$(CC) $(CFLAGS) -c $<

clean:
	rm -f $(TARGETS) $(OBJS) simon.o

format:
	clang-format -i -style=file *.[ch]
```