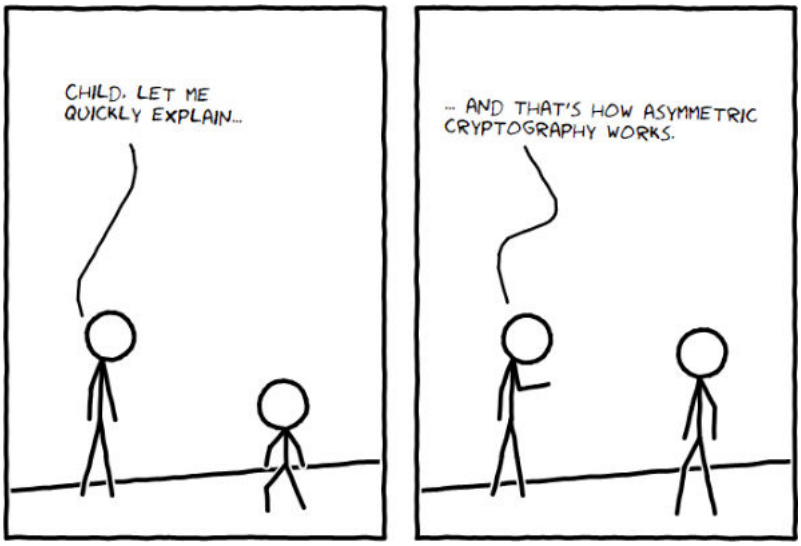
Cryptography Design

Ralph Miller

November 21st 2021



"Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin."

John von Neumann

Contents

1 Objective

In this assignment I will create three executable programs that will encrypt files using RSA public and private key pairs. These programs will be supported by two other function containing programs, rsa.c and numtheory.c. The three programs are as follows. Keygen, a program to create public and private RSA key pairs used in Encryption/Decryption. Encrypt, a program that takes the RSA public key and a input file and encrypts the input file. Decrypt, a program that uses the RSA private key and the encrypted infile to decrypt the message and write the message to outfile.

1.1 Deliverables

```
decrypt.c   : This contains the implementation and main() function for the decrypt program.
encrypt.c   : This contains the implementation and main() function for the encrypt program.
keygen.c    : This contains the implementation and main() function for the keygen program.
numtheory.c : This contains the implementations of the number theory functions.
numtheory.h : This specifies the interface for the number theory functions.
randstate.c : This contains the implementation of the random state interface for the RSA library
and number theory functions.
randstate.c : This specifies the interface for initializing and clearing the random state.
rsa.c       : This contains the implementation of the RSA library.
rsa.h       : This specifies the interface for the RSA library.
Makefile    : This creates all binaries.
README.md   : Contains information about the program and how to use it.
```

2 RSA

The RSA functions contain the functions necessary for generating, writing, reading, and verifying RSA keys. These functions are supported by the Number Theoretic functions in numtheoery.c. The RSA functions will make a private and public key, using two large prime numbers q, a public exponent e, and a public modulus n. These keys are written to their appropriate files, rsa.pub and rsa.priv which are used in the Encryption and Decryption modules.

3 Keygen

The key generator in this program generates a 256 RSA key using custom gmp and RSA functions. First, it parses the command line options using getopt(). Then it opens the public and private key files using low level system calls. It then sets the permissions for the private key file to read/write for user only. Next it initializes the gmp random state using the set seed. Once this is done, it creates the public and private keys using the written RSA functions. Finally it gets the user's name as a string and converts it to an mpz type and writes the public and private keys to the files. If verbose printing is enabled, the following is printed.

1. username
2. signature 's'
3. 1st large prime 'q'
4. 2nd large prime 'q'
5. public modulus 'n'
6. public exponent 'e'

7. private key 'd'
8. all mpz type variables with num of bits

3.1 Command Line Options

```
-b        : specifies the minimum bits needed for the public modulus n.
-i        : specifies the number of Miller-Rabin iterations for testing primes (default: 50).
-n pbfile : specifies the public key file (default: rsa.pub).
-d pvfile : specifies the private key file (default: rsa.priv).
-s        : specifies the random seed for the random state initialization
            (default: the seconds since the UNIX epoch, given by time(NULL)).
-v        : enables verbose output.
-h        : displays program synopsis and usage.
```

4 Encrypt

The encryption program first parses command line options using getopt(). Next it opens the public key file using low level system calls, grabbing the public key. If verbose printing is enabled it will print the username, signature 's', public modulus 'n', public exponent 'e', and all mpz type variables and the num of bits. Then it will convert the read in username into an mpz type. This is the expected value of the verified signature. The signature is then verified and encrypted with RSA functions. Lastly the public key file is closed and mpz type variables are cleared.

4.1 Command Line Options

```
-i : specifies the input file to encrypt (default: stdin).
-o : specifies the output file to encrypt (default: stdout).
-n : specifies the file containing the public key (default: rsa.pub).
-v : enables verbose output.
-h : displays program synopsis and usage
```

5 Decrypt

The decryption program first parses the command line options using getopt(). Next it opens the private key file using stdio open functions. Grabbing all of the variables encrypted. If verbose output is enabled print the public modulus 'n' and the private key 'e'. Then the file is decrypted using the rsa decrypt file function. Lastly all mpz type variables are cleared and files are closed.

5.1 Command Line Options

```
-i : specifies the input file to decrypt (default: stdin).
-o : specifies the output file to decrypt (default: stdout).
-n : specifies the file containing the private key (default: rsa.priv).
-v : enables verbose output.
-h : displays program synopsis and usage.
```