Team 1 EN.605.204.81 ARM32 RSA Design Document V2

Rohan Abraham, Tero Suontaka, Sullivan Prellwitz

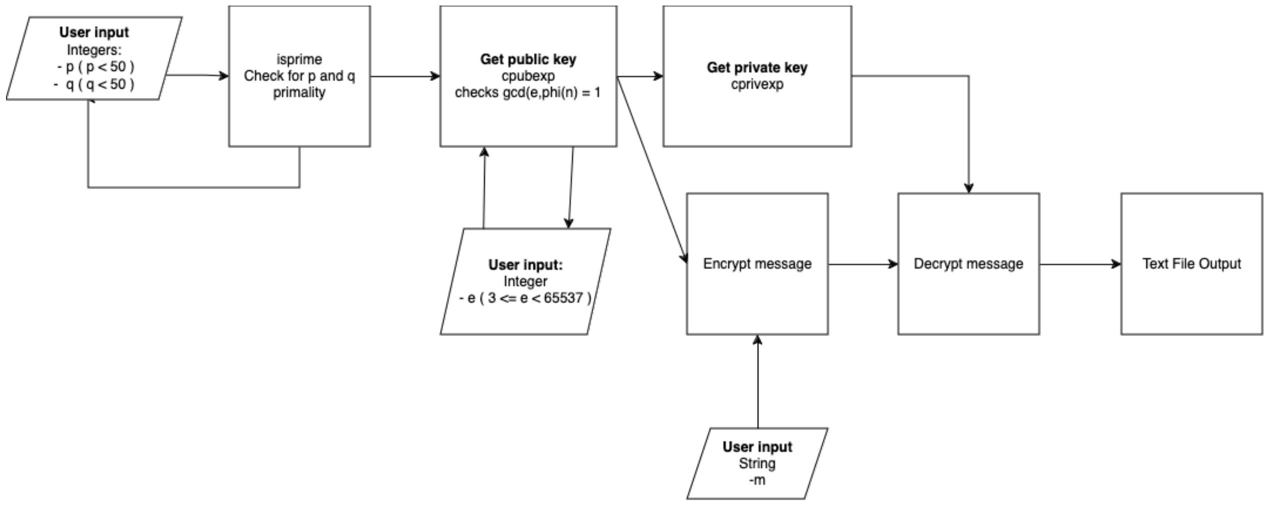
April 28, 2024

Contents

1 Goals

Purpose: Encrypt and decrypt messages using a custom RSA implementation in ARM32 assembly. Implement a modular design for all functions and create a library of assembly code that enables the generation of a public and private RSA keys using user specified values.

2 Architecture



3 Functions

3.1 libIO.s

3.1.1 stringToArray

Purpose: Converts a string (byte array) to an array of 32 bit integers
input:

- r0 - pointer to string
- r1 - size of string

Output:

- r0 - pointer to integer
- r1 - size of array

3.1.2 arrayToString

Purpose: Converts an integer array to a null delimited string
input:

- r0 - pointer to integer array
- r1 - size of array

Output:

- r0 - pointer to string
- r1 - size of string

3.1.3 writeFile

Purpose: Write to a file, name provided by user
input:

- r0 - name of file to write
- r1 - pointer to message to write

3.1.4 writeArray

Purpose: Write 32 bit integer array to a file
input:

- r0 - pointer to string
- r1 - pointer to message to write
- r2 - length of string

3.1.5 readArray

Purpose: Read file to 32 bit integer array
input:

- r0 - name of file to read

Output:

- r0 - pointer to array
- r1 - array length

3.2 libMath.s

3.2.1 gcd

Purpose: Computes the greatest common divisor of two integers
input:

- r0 - first integer to compute gcd of
- r1 - second integer to compute gcd of

Output:

- r0 - greatest common divisor of two input integers

3.2.2 mod

Purpose: Modulo calculation: r0 mod r1 = r0
input:

- r0 - first integer to compute modulo
- r1 - second integer to compute modulo

Output:

- r0 - modulo value

3.2.3 isPrime

Purpose: Determines if a number is prime
input:

- r0 - integer to test

Output:

- r0 - binary value indicating primality returns -1 for invalid values

3.2.4 totient

Purpose: Totient calculation $\Phi(n) = (p - 1)(q - 1)$ s.t. p and q are prime
input:

- r0 - p
- r1 - q

Output:

- r0 - return: totient value of (n) or r0 == -1 if p or q are NOT prime (error)

3.3 libRSA.s

3.3.1 cprivexp

Purpose: Calculates the private exponent. Calculates multiplicative inverse of public key over ring of integers mod n
input:

- r0 - public exponent (e)
- r1 - integer such that gcd(r0,r1) = 1 (phi(n))

Output:

- r0 - private exponent returns -1 if gcd(r0,r1) != 1

3.3.2 cpubexp

Purpose: Validates the public exponent s.t. $1 < e < \Phi(n)$ and e is co-prime to $\Phi(n)$ [$\gcd(e, \Phi(n)) = 1$]
input:

- r0 - p
- r1 - q
- r2 - e

Output:

- r0 - pub exponent or -1 if error

3.3.3 process

Purpose: Processes the input for RSA encryption and decryption. For encryption, use private key as exponent. For decryption, use public key as exponent
input:

- r0 - integer base a
- r1 - integer exponent b
- r2 - integer modulus n

Output:

- r0 - a ^ b mod n

3.3.4 processArray

Purpose: Processes an integer array for RSA encryption and decryption. Applies a^b mod n for all a in array.
input:

- r0 - pointer to integer array
- r1 - size of array
- r2 - integer exponent b
- r3 - integer modulus n

Output:

- r0 - pointer to processed integer array
- r1 - size of array

3.3.5 generateKeys

Purpose: Prompt user for primes and public exponent and generate private key

3.3.6 encrypt

Purpose: Encrypts a message given user input public key and modulus and writes to encrypted.txt

3.3.7 decrypt

Purpose: Decrypts a message from encrypted.txt given user input private key and modulus and writes plaintext to plaintext.txt

3.4 main.s

3.4.1 main

Purpose: Drives the generation of keys, encryption, and decryption

4 Testability

The individual components of the RSA implementation will be tested using a test script per function. These test scripts will cover normal, edge, and absurd cases in order to ensure proper functionality. The test scripts will be able to be run all at once using a control script.
Test scripts will be help within a `/tests` folder within the project repository and will take the form

```
{function_name}-tests.{file_extension}
```

5 Timeline

5.1 March 11 - 15

- First implementation meeting
- Initialize code repository
- mod function implementation finished, tests written

5.2 March 25 - 29

- Second implementation meeting
- gcd, pow, and tot implementation finished, tests written
- Plan next implementation steps

5.3 April 8 - 20

- Meet as needed
- RSA implementation finished (April 20), tests written
- Creation of testing control script

5.4 April 21 - 27

- Complete testing
- Squash bugs
- Prep repository and extra materials for submission

5.5 April 28

- Submit implementation