Krisha Sharma

CSE13s

Winter 2022 Long

02/04/2022

Assignment 5 DESIGN.pdf

Description of Program:

The main task in Assignment 5: Public Key Cryptography is to write a function that allows you to tell whether or not a large number is a prime number.

Files to be included in directory "asgn5":

- `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 implementation of the number theory functions
- `randstate.c`
    - This contains the implementation of the random state interface for the RSA library and number theory functions
- `randstate.h`
    - This specifices the interface for initalizing and clearing the random state

- `rsa.c`
    - This contains the implementation of the RSA library
- `rsa.h`
    - This specifices the interface for the RSA library
- Makefile
    - `CC = clang` must be specified
    - `CFLAGS = -Wall -Wextra -Werror -Wpedantic` must be specified
    - `pkg-config` to locate compilation and include flags for the GMP library must be used.
    - `make` must build the `encrypt, decrypt,` and `keygen` executable, as it should `make all.`
    - `make decrypt` should build only the `decrypt` program.
    - `make encrypt` should build only the `encrypt` program.
    - `make keygen` should build only the `keygen` program.
    - `make clean` must remove all files that are compiler generated.
    - `make format` should format all the source code, including the header files
- README.md
    - Text file in Markdown format that describes how to build and run the program, how the program handles erroneous inputs, and any problems encountered while developing the program.
- DESIGN.pdf
    - Describes design for the program thoroughly with pseudocode and visualizations.
- WRITEUP.pdf

- This document must be a PDF
- What did you learn from the different sorting algorithms? Under what conditions do sorts perform well? Under what conditions do sorts perform poorly? What conclusions can you make from your findings?
- Graphs explaining the performance of the sorts on a variety of inputs, such as arrays in reverse order, arrays with a small number of elements, and arrays with a large number of elements. The graphs must be produced using either `gnuplot` or `matplotlib`.
- Analysis of the graphs produced.

General Notes:

- Assignment 5: Public Key Cyptrography is focused around RSA algorithms.
- We are learning this in order to create an RSA key pair, similar to what ssh does when prompted ssh keygen.
  - To create an RSA key pair (mathematics of RSA):
    - 1. Choose two large prime numbers p and q
    - 2. Compute the product of the two, aka public modulus, $n = p \cdot q$
    - 3. Compute d and e, where $d \cdot e \equiv 1 \; mod(\phi(n))$
      - $\phi(n) = \phi(p) \cdot \phi(q)$

        $= (p - 1) \cdot (q - 1)$
      - $\phi(p)$ = # of numbers < p that are coprime w/ p
        - coprime meaning that if you do $gcd(x, p) = 1$
      - In this case d and e are multplicative inverses (modular inverses) in a modular ring

- In order to guarantee that there is a modualo inverse of e, you need to make sure that the $gcd(e,\ \phi(n))\ =\ 1$
    - This also guarantees an inverse d

- To encrypt and decrypt a message:
    - encryption $c\ =\ m^{e}(mod(n))$
    - decryption $m\ =\ c^{d}(mod(n))$
        - In this case m is some message
            - For example: given “abc” which is 3 bytes, when converted into is ASCII value it becomes [97, 98, 99]. That then becomes some long binary (1100010…. …… ….) that is evaluated as a single number, which is what m is, that is then raised to e, becoming $m^{e}(mod(n))$.
        - n is the public modulus
    - Rather than using ϕ, during this assignment we are instead using lamda λ, which is Carmichale’s quotient