Assignment-RSA implementation in C

(Cryptography & Security Implementation)

Souvik Santra(CrS 2408)

1.Solutions:

Two programms have been attached with it. Once the two primes are of 512 bits, each. After that one prime is of 768 bit, another is of 1024 bit.

The following steps have been followed-

i) Prime Generation : Generate two primes of specific bits. The bits are choen uniformly at random using `mpz_urandomb.`

ii) Finding Totient function : Computes the produvt of two primes as '*N*, the Eulers' phi function is *phi(N)=(p-1)(q-1).*

iii) Key generation: We have used one of the commonly known public exponent *e=65537.*

Compute the private key *'d'* (the multiplicative inverse of *e*) using `mpz_invert.`

iv) Enc & Dec : Encryption & Decryption have been done & verified that dycrypted message is exactly same as the original plain text.

v) Clock cycle calculation : We have used `rdtsc` for measuring the clock cycle fo each operation.

vi) Result(output) : Minimum, maximum, average clock cycles for each operation.

2. The clock cycle measurement table:

A) When the two primes are of 512 bit each :

(For 10000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*1133328*	*40244957*	*3264389.88*
Second prime generation	*1131816*	*43562618*	*3265428.54*
Computing '*N*'	*168*	*23436*	*473.21*
Computing *phi*	*252*	*177436*	*567.90*

Clock cycles for computing *'d' : 4396*

Clock cycle for Encryption: 19068

Clock cycle for Decryption: 242928

(For 50000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*1139628*	*47239976*	*3184736.67*
Second prime generation	*1133888*	*39302508*	*3183564.31*
Computing '*N*'	*168*	*261716*	*459*
Computing *phi*	*252*	*156800*	*511.69*

Clock cycles for computing *'d' : 3976*

Clock cycle for Encryption: 17164

Clock cycle for Decryption: 239232

(For 100000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*1128288*	*22509760*	*3122441.76*
Second prime generation	*1130780*	*32031131*	*3115301.68*
Computing '*N*'	*168*	*207088*	*381.02*
Computing *phi*	*252*	*145320*	*437.77*

Clock cycles for computing *'d': 2352*

Clock cycle for Encryption: 18004

Clock cycle for Decryption: *238700*

(For 1000000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*1129184*	*47305804*	*3145238.24*
Second prime generation	*1126888*	*44003540*	*3143621.52*
Computing '*N*'	*168*	*2151800*	*420.48*
Computing *phi*	*252*	*1720852*	*487.55*

Clock cycles for computing *'d': 7588*

Clock cycle for Encryption: 18116

Clock cycle for Decryption: *262864*

B) When the first prime is of 768 bit & the second one is of 1024 bit:

(For 1000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*3117072*	*51576868*	*10695239.32*
Second prime generation	*6514928*	*156810136*	*28512964.29*
Computing '*N*'	*476*	*10248*	*1313.42*
Computing *phi*	*560*	*5740*	*1123.72*

Clock cycles for computing *'d' : 8176*

Clock cycle for Encryption: *109872*

Clock cycle for Decryption: *1238272*

(For 10000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*3118136*	*80518838*	*10935252.79*
Second prime generation	*6460832*	*230366529*	*28203228.76*
Computing '*N*'	*448*	*54404*	*1194.88*
Computing *phi*	*532*	*127988*	*1264.87*

Clock cycles for computing *'d' : 10248*

Clock cycle for Encryption: *108836*

Clock cycle for Decryption: *1203160*

(For 50000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*3114832*	*285149844*	*11488944.51*
Second prime generation	*6463996*	*776917538*	*29629981.53*
Computing '*N*'	*392*	*715596*	*1431.40*
Computing *phi*	*504*	*472136*	*1531.95*

Clock cycles for computing *'d' : 12180*

Clock cycle for Encryption: *448224*

Clock cycle for Decryption: *1459360*

(For 100000 trials)

Operation	Min clk cycle	Max clk cycle	Avg clk cycle
First prime generation	*3136280*	*115129080*	*110773197.14*
Second prime generation	*6497512*	*297725904*	*28580636.04*
Computing '*N*'	*420*	*151900*	*1476.86*
Computing *phi*	*504*	*200312*	*1587.26*

Clock cycles for computing *'d' : 18284*

Clock cycle for Encryption: *303268*

Clock cycle for Decryption: *1440012*

3.) System Specification:

CPU: AMD Ryzen 5 7520U with Radeon Graphics(Cores: 4 | Threads: 8 | Arch: x86_64)

RAM Size: 2075 GB

Operating System : Ubuntu 24.04.3 LTS

GCC Version: gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

Library used: GNU Multiple Precision Arithmetic Library(GMP)

4) Observation about the results(outputs):

i) Prime generation: The average clock cycles increase with the key size. The large gap between minimum clock cycle & maximum clock cycles indicates the variability in the number of iterations needed for finding primes using `mpz_nextprime.`

ii) Phi function: the clock cycle for calculation Phi function is very very fast compare to the clock cycles for finding primes. Although, when the primes get bigger, naturally the clock cycles is increasing.

iii) Key generation('*d*') : The clock cycles increases when the prime size increses. In my observation, while performing RSA taking 512bit primes, the key generation clock cycles decreses when number of trials increases(from 10k to 1lakh). But for 1 million trials, this clock cycles is a little bit high compare to the other. When, we have took 768 bit & 1024 bit primes, the clock cycles for findind '*d*' is much high than the previos case.

iv)Encryption & Decryption: Always we have less clock cycle for encryption than the clock cycle for decryption. We, are getting success for every decryption scenario, by checking that the original plain text is same as decrypted message.

v) Dependency on CPU(system): The output i.e. the number clock cycles for every case & for each operation, it depends on the system characteristics. The updated gcc version may affect in the optimizing performance.

5)References :

i) Open ChatGPT : taken suggestion for appropriate code & its modification, usage of some functions(that is used in programm).

ii)Website: www.geeksforgeeks.org

iii)The paper, " RSA Cryptography Algorithm & its Applications to Security System by Using Linear Congruence: by Souad Mugassabi"