CRYPTOGRAPHY AND COMPUTER SECURITY || COL759

Assignment - 2
Implementing the Quadratic Sieve Algorithm

Surajprakash Narwariya [2024JCS2044]
Anand Sharma [2024JCS2049]

Objective

The objective of this assignment is to implement the Quadratic Sieve Algorithm for integer factorization. This report gives the detailed approach which we used for factorization. We have used GMP (GNU Multiple Precision Arithmetic Library) to handle large numbers and threading to parallelize the sieving step. It was the most time consuming part of the program. We followed the steps as described by sir in the slides.

Steps Followed

- ➔ We have used the GMP library for handling large numbers.
- ➔ Factor Base Size Selection: Choosing appropriate factor base is a crucial part for the Quadratic Sieve Algorithm. We estimated the Factor Base Size B as follows:

$$B = \exp(0.474 * \sqrt{\log(n) * (\log(\log(n)))})$$

- ➔ After calculation of the factor base size, we implemented Sieve of Eratosthenes for getting a list of prime numbers. Then, we searched for the prime numbers to be included in the factor base. The factor base is generated by finding small primes. Only those prime numbers(p) are added in the factor base for which the legendre symbol is 1. [**(n/p) = 1**]
- ➔ The calculation of the legendre symbol is done using a function of the GMP library namely, mpz_legendre(n,p).
- ➔ After getting the factor base, we then calculated m, where m= squareRoot(n).
- ➔ The sieving method is multi-threaded, dividing the range into sub-ranges for each thread. Each thread searches for B-smooth numbers in its assigned range and

stores the factorization exponents in the matrix. The calculation of the sieving interval size is done using the formula, range = B^2.

- ➔ We put the x values ranging from -(range/2) to +(range/2),
- ➔ Then, we started the sieving process on ‘a’ different threads. On receiving each Q(x), we checked for it to be B-smooth. If the number is B-smooth, we create the 0/1 vector and add it to the matrix. We are taking the total (t+200)[t=size of factor base] number of B-smooth values in the matrix.
- ➔ After the creation of the 0/1 matrix. We Implement Gaussian Elimination on the matrix. The matrix is reduced using Gaussian elimination, which helps in identifying the linear dependencies required for factorization. After implementing the Gaussian elimination, we find the linear dependency from the matrix.
- ➔ Using the linear dependencies, we find the values for which the linear dependencies came out to be 0. Consider these values, we find the values of x and y. Now using these values we check whether two non-trivial factors exist. If it exists, we return the value.
- ➔ Once a dependency (a solution to the linear combination modulo 2) is found, the idea is to use the corresponding rows to compute x and y. The goal is to use the following congruence relation to factor the number n: x^2 congruent to y^2 mod n. This implies: (x−y)(x+y) congruent to 0(mod n)
- ➔ If both x and y are non-trivially congruent which means, neither x=y(mod n) nor x=−y(mod n), then taking the greatest common divisor (gcd) of x−y and n, and x+y and n, gives a non-trivial factor of n.

How to Run the Program and Test

- ➔ Insert the number to be tested in the main.cpp file on line number 47.
- ➔ Command to run the program:
  - ◆ make [Compiles the program]
  - ◆ make clean
  - ◆ make run

NOTE: You need to specify your library path in the make file for the compilation