COMP4920 Senior Design Project 2, Spring 2020
Advisor: Dr. Hüseyin Hışıl

FIWE: Factoring Integers with ECM Requirements Specification Document

24.05.2020
Revision 3.0

By:
Asena DURUKAN
Student ID Number: 16070001016
Aslı ALTIPARMAK
Student ID Number: 15070001003
Elif ÖZBAY
Student ID Number: Ç15070002011
Hasan Ozan SOĞUKPINAR
Student ID Number: 17070001047
Nuri Furkan PALA
Student ID Number: 15070006030

Revision History

Revision	Date	Explanation
1.0	26.11.2019	Initial Requirements
2.0	28.12.2019	Requirements Model
3.0	24.05.2020	Projective and affine coordinate comparison is removed Group law example is converted to projective coordinates ECM pseudocode is modified Usability requirement is added

Contents

1 Introduction

Lenstra's Elliptic Curve Method (ECM) is one of the algorithms that are proposed as a solution to the factorization of integers problem which is commonly used in asymmetric cryptosystems. The method simply does arithmetic operations with the points on an elliptic curve and finds a factor of the integer that the curve is defined over. The aim of this project is to implement the ECM algorithm on GPU's Single Instruction Multiple Data (SIMD) architecture by parallelizing the algorithm.

1.1 Background

1.1.1 Factorization of Integers

Factorization of composite integers is a difficult problem which has no known algorithm that works in polynomial time. The difficulty of prime factorization makes it a valuable problem in cryptology.
Many cryptosystems are constructed upon a one-way function which means fast to solve in one-way and slow in the other way, such as factorization. It is quite simple to multiplicate 5 with 17. However, it takes a significantly longer time to find the prime factors of 85. This computation time increases exponentially with respect to the number being factorized when the composite number is chosen larger in size.
Since this problem is used as a basis of common cryptosystems, many studies have been conducted to figure out an algorithm that factorizes integers as fast as it can be. The most famous algorithms are briefly explained in Section 1.1.2.

1.1.2 Factorization Methods

Factorization methods are usually categorized into two classes which are special and general algorithms. Special algorithms are applicable to a specific group of numbers while general algorithms are appropriate for all composite numbers [3].

Common Special Algorithms
Pollard's $p-1$ method is an algorithm that relies on *Fermat's Little Theorem* which states that $a^{p-1} \equiv 1 \pmod{p}$ where p is prime and $gcd(a,p) \equiv 1$. The performance of the algorithm depends strictly on the selection of a. Moreover, the method works efficiently if $p-1$ is $B-smooth$ for an arbitrarily chosen B (composed of small primes until B), where p is a prime factor of the number to be factorized.
Pollard's ρ method is another special algorithm that is based upon the birthday paradox which says that two persons have the same birthday if they are chosen from a group of at least 23 people, with the probability of 50%.
Elliptic Curve Method is a derivation of the $p-1$ *method* which replaces the multiplicative group with the group of points on an elliptic curve. This modification overcomes the disadvantage of the $p-1$ *method* that requires the smoothness of $p-1$ [5].

Common General Algorithms
Congruent Squares is based on the equation $x^2 - y^2 = (x+y)(x-y)$. If n is a multiple of $x^2 - y^2$, then the factors of n must divide $x+y$ and $x-y$ [3].
Quadratic Sieve tries to find x and y values such that $x^2 \equiv y^2 \pmod{n}$ and in case of success, applies *Euclidean Algorithm* for the probability of a nontrivial factor [4].
Number Field Sieve is a method similar to *Quadratic Sieve* but combined with algebraic number fields. The difference is the field that is used [3].

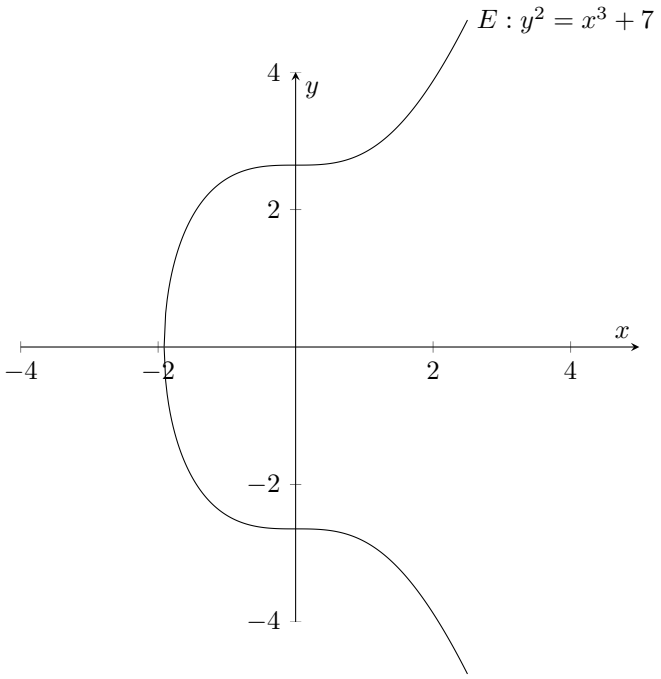

Figure 1: Elliptic Curve

1.1.3 Elliptic Curves

An elliptic curve is a set of points that lies in the equation

$$y^2 = x^3 + Ax^2 + B \tag{1}$$

called *Short Weierstrass Equation* where A and B are constants, together with a hypothetical point ∞. The x and y values that satisfy the equation represent the points on the curve defined. The demonstration of an elliptic curve is shown in Figure 1. A curve can be defined over a group, ring, or field. Defining an elliptic curve over a finite structure makes the calculations easier.
The points can be represented as x,y pairs which are called the affine representation. With a simple map, the points can be represented in projective coordinates by adding the third parameter Z, such as X,Y,Z. A point can be represented in both ways, and the mapping from one to the other is always possible.

The Group Law
Since an elliptic curve is a group, the addition operation must be defined. Let $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$ and $P_3 = P_1 + P_2$. Also, assume a point $P_d = (X_d, Y_d, Z_d)$ is the difference of P_1 and P_2.

$$\begin{aligned} X_3 &= Z_d * ((X_1 - Z_1) * (X_2 + Z_2) + (X_1 + Z_1) * (X_2 - Z_2))^2 \\ Z_3 &= X_d * ((X_1 - Z_1) * (X_2 + Z_2) - (X_1 + Z_1) * (X_2 - Z_2))^2 \end{aligned} \tag{2}$$

Addition of two points P_1 and P_2 in projective coordinates is declared algebraically in Equation 2 [2]. It can also be described geometrically. Geometrically, the addition operation has two steps. In the first step, the line that goes through the points P_1 and P_2 is calculated. That line certainly intersects the curve on a third point. In the second step, the symmetric of that calculated point is taken, according to the x-axis. That point P_3 is the result of the addition.
Doubling is similar to the addition operation. To find the line that goes through the point, its derivative is taken. The second step is same with the addition.

1.1.4 Elliptic Curve Method

The method aims to find the factors of an integer n greater than 1. The algorithm is explained below.

Algorithm 1: Elliptic Curve Method Algorithm

```
   input  : n: composite number to be factorized
   output: d: factor of n
 1 Generate a random projective curve E and a point P on it
 2 Choose an integer k that is composed of small prime factors
 3 Compute k * P on E
 4 if k * P failed due to the lack of inverse of an element then
 5 |  return d = gcd(k, n);
 6 else
 7 |  if Threshold for k has not been exceeded then
 8 |  |  Go back to Step 2.
 9 |  else
10 |  |  if Threshold for E has not been exceeded then
11 |  |  |  Go back to Step 1.
12 |  |  else
13 |  |  |  Failed. Stop.
14 |  |  end
15 |  end
16 end
```

1.1.5 Parallelization of the Algorithm

Parallel computing is the simultaneous use of multiple computation resources to solve a computational problem. In other words, it is the process of large computations which can be broken down into multiple processors using multiple threads that can process independently and whose results are combined upon the completion [6]. A key problem of parallelism is to reduce data dependencies in order to be able to perform computations on independent computation units with minimal communication between them. However, on the SIMD architecture, data dependency is not a problem to be considered since each data is processed by an identical set of instructions.
The aim of this project is to parallelize the mathematical operations of the ECM algorithm on the CUDA platform.

1.1.6 Safegcd

A Greatest Common Divisor (GCD) algorithm is needed in the ECM algorithm. GCD is a concept in number theory defining the greatest of the integers that divides each operand. Generally, it is represented as $gcd(a, b)$ where a and b are positive integers.
Modular inversion operation is also a part of the ECM algorithm. Basically, the formula of modular multiplicative inversion is $a \times a^{-1} = 1 \pmod{n}$ where a^{-1} is the modular inverse of a in mod n.

An algorithm called *safegcd* is announced recently which is a variation of the binary GCD algorithm [1]. This algorithm is fast and it can be easily converted to a constant time algorithm, which is essential in crypto since non-constant time algorithms cause security vulnerabilities. Additionally, the method has the potential to be parallelized and speeded up which is this project seeks for. The safegcd algorithm is implemented within the scope of this project. It is chosen due to its speed, recency and the potential to be parallelized. One of the parts is for integer inputs, and the other is for polynomial inputs [1]. Implementing the algorithm for integer inputs, which is given in Algorithm 2 is enough for the ECM algorithm.

Algorithm 2: Safegcd Algorithm

input : f, g, *precomp*, t
output: z
Let $mul(a, b, c, d)$ be $a \times b + c \times d$
Initialize f as p
Initialize δ, U, R as 1
Initialize V, Q, i as 0
while $i < 12$ **do**
    Imitate division on δ, f, g and assign back to δ, u, v, q, r
    Assign $mul(u, f, v, g)$ to f
    Assign $mul(q, f, r, g)$ to g
    Assign $mul(u, U, v, Q)$ to U
    Assign $mul(q, U, r, Q)$ to Q
    Assign $mul(u, V, v, R)$ to V
    Assign $mul(q, V, r, R)$ to R
    Shrink f and g
    Increment i
end
Assign $f^{-1} \pmod{n}$ to z

2 Requirements List

2.1 Functional Requirements

2.1.1 Mathematical Requirements

Arithmetic

- Storing multi-precision integers
- Arithmetic operations on multi-precision integers

Elliptic Curves

- Defining and storing elliptic curves
- Arithmetic operations on elliptic curves

Greatest Common Divisor

- Calculating the GCD of two integers

Integer Modular Inversion

- Calculating the modular inverse of an integer

2.1.2 Parallelization Requirement

- To be able to factorize multiple composite numbers simultaneously

2.1.3 Usability Requirement

- A user interface which user can factorize a composite number simply

2.2 Non-functional Requirement

- The GCD must have a small time complexity

3 Actors and Use Cases

The only purpose of the software is factoring integers. The user is going to enter the large composite number as the input and get the factors of it as the output. The use case diagram of the software is shown in Figure 2.

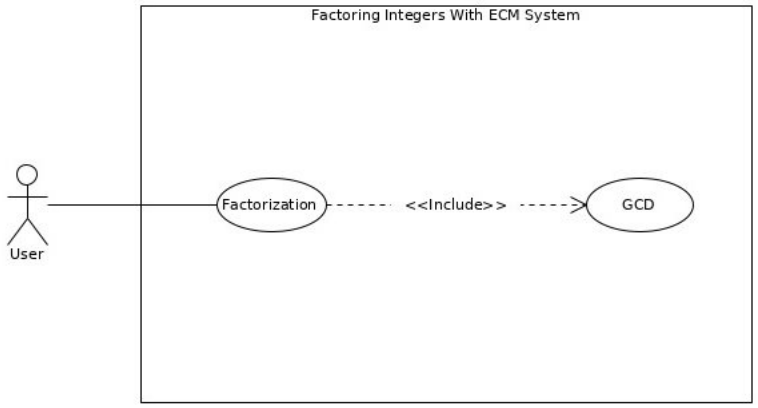

Figure 2: Use Case of the Software

4 Glossary

In this paper, some notations are used to eliminate repetitions. The notations are listed below. Unless otherwise stated, they are going to refer to the following descriptions in the list.

- B: An integer that composed of small primes
- CUDA: Computer Unified Device Architecture
- E: Elliptic curve
- ECM: Elliptic Curve Method
- GCD: Greatest Common Divisor
- GPU: Graphics Processing Unit
- n: The composite number to be factorized
- $P = (x, y)$: A point on the curve
- SIMD: Single Instruction Multiple Data
- ∞: Point at infinity on E

References

[1] D. J. Bernstein and B. Yang. Fast constant-time gcd computation and modular inversion. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):340–398, 2019.

[2] H. Hışıl. *Elliptic Curves, Group Law and Efficient Computation.* PhD thesis, Queensland University of Technology, 4 2010.

[3] P. L. Jensen. Integer factorization. Master's thesis, University of Copenhagen, 2005.

[4] E. Landquist. The quadratic sieve factoring algorithm. 01 2001.

[5] H. W. Lenstra. Factoring integers with elliptic curves. volume 126, pages 649–673. Annals of Mathematics, 1987.

[6] D. Thakur. What is parallel computing? - definition.