FIWE: Factoring Integers with ECM

Asena Durukan, Aslı Altıparmak, Elif Özbay, Hasan Ozan Soğukpınar, Nuri Furkan Pala
Supervisor: Dr. Hüseyin Hışıl

Aim of the Project

Lenstra's Elliptic Curve Method (ECM) is one of the algorithms that is proposed as a solution to the factorization of integers problemwhich is commonly used in asymmetric cryptosystems. ECM is an algorithm that factorizes an integer by using the properties of elliptic curve arithmetic. The aim of this project is to implement ECM on Single Instruction Multiple Data (SIMD) processors in a parallelized manner and to implement safegcd algorithm. The implementation is done in projective space using Montgomery curves for the sake of performance.

ECM

Algorithm 1: ECM Algorithm

input : n: composite number to be factorized
output: d: factor of n

Generate a random projective curve C and a point P on it
Choose an integer k that is composed of small prime factors
Compute $k * P$ on C
if $k * P$ *failed due to the lack of inverse of an element* **then**
    return $d = gcd(k, n)$;
else
    if *Threshold for* k *has not been exceeded* **then**
        Go back to Step 2.
    else
        if *Threshold for* C *has not been exceeded* **then**
            Go back to Step 1.
        else
            Failed. Stop.
        end
    end
end

Safegcd

The ECM algorithm needs Greatest Common Divisor (GCD) and modular inversion operations. For this purpose, the binary GCD and safegcd algorithms are coded. The reason behind the safegcd decision is based on its speed and recency. Moreover, it can be easily converted to a constant time algorithm, and also can be parallelized. Even though there are two main parts of the safegcd algorithm, the algorithm for integer inputs is suitable for the ECM implementation [1]. The binary GCD algorithm is used inside ECM and the integration of safegcd to the software is left as a future work.

Parallelization

Parallelization is a process of breaking down large computations into multiple processors [2]. 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. In this project, the parallelization of the ECM algorithm is done on the SIMD architecture of GPU.

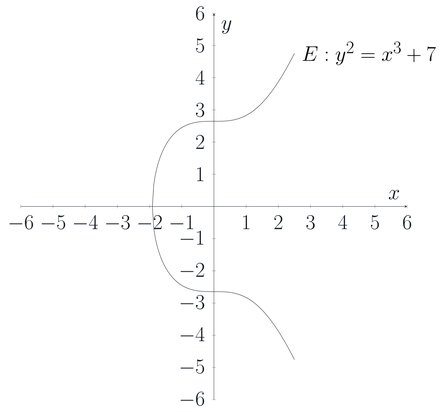

Figure 1 Elliptic Curve

Test & Operation

For all functions in the Montgomery and ECM libraries, test cases are generated by the help of GMP and Magma. While GMP checks the correctness of the results in run-time, Magma tests are handled with file read and writes. The results of the tests are listed in the table below.

Table 1 Test Statistics

Function Name	Is Tested?	Size of Inputs	# of Trial	# of Success	# of Fail	Time Spent in C (sec)
proCurvePointGMP	Yes	1-100	10000	10000	0	15
proAddGMP	Yes	1-100	10000	10000	0	12
proAddMagma	Yes	1-100	10000	10000	0	45
proDblMagma	Yes	1-100	10000	10000	0	33
proLadderGMP	Yes	1-100	10000	10000	0	27
proLadderMagma	Yes	1-100	10000	10000	0	143
safegcdPyMul	Yes	1-100	10000	10000	0	37
ecmGmpTest	Yes	1-10	1000	1000	0	34

There are installation and operation processes for the FIWE project as all qualified software projects. Users can use FIWE in 2 different ways which are listed below.

- Source code as a library
- Bash script on Unix-like terminal

Installation is needed to use the Bash script. Usage of the Bash Script is explained in Table 2.

Table 2 Commands & Explanations

Command	Explanation
fiwecm - -help	Shows the manual
fiwecm <number>	Printouts a factor of the number
fiwecm -f <in.txt>	Printouts the factors of the numbers inside "in.txt"
fiwecm <number> -o <out.txt>	Writes a factor of the number to "out.txt"
fiwecm -f <in.txt> -o <out.txt>	Writes the factors of the numbers inside "in.txt" to "out.txt"

Conclusion

Within the scope of this project, the ECM algorithm is implemented using the Montgomery curves on projective coordinates. Also, the safegcd algorithm is implemented to be run on CPU. Finally, the ECM implementation is parallelized on the SIMD architecture of GPU.

Benefits of the Project

- Provides a resource that can guide people who study in the fields of cryptology, cryptanalysis and some other related subjects.
- Anyone can understand and use this software by reading the reports and examining the source codes.
- Provided the group members to understand the integer factorization problem and a solution to this problem, no matter how professional it is, and to have a basic knowledge of cryptology and cryptanalysis.
- Helps users understand this concept and implement it on their own work.

Future Work

- The ECM algorithm has a second phase which aims to find the factor that first phase could not found. Second phase can be implemented as a future work.
- The integration of the safegcd algorithm to the ECM algorithm is possible.
- The ECM algorithm is implemented using projective coordinates, however affine coordinates can also be used. A comparison can be made between the performances of ECM using projective and affine coordinates.
- For the parallelization part, the software can be implemented on brand new GPUs and CPUs or with different architectures.

Acknowledgements

We would like to thank our academic advisor Dr. Hüseyin Hışıl for his great support and encouragement in this project. He guided us and helped us see it is not as complicated as it seems. Besides our advisor, we would like to thank Dr. Mutlu Beyazıt who conducted this lecture without any ambiguity.

References

[1] Daniel J. Bernstein and Bo-Yin Yang. "Fast constant-time gcd computation and modular inversion". In: 2019 (May 2019), pp. 340–398.doi:10.13154/tches.v2019.i3.340-398.
[2] Dinesh Thakur. What is Parallel Computing? - Definition