Optimization and Implementation of various algorithms to speed up scalar multiplication over Elliptic Curves

Prathamesh Dhake 190070048
Atharva Raut 190070050
Jaideep Chawla 190110030

March 20, 2023

1 Abstract

Elliptic Curves form an important basis for existing encryption algorithms. It generates security between key pairs for public key encryption by using the mathematics of elliptic curves. All elliptic curve cryptography is based on the difficulty of finding a secret integer n given the scalar multiple $Q = [n]P = P + \cdots + P$ of a base point P on an elliptic curve, say E/k where k is the prime field.

An example would be the curve **Curve25519** represented as $y^2 = x^3 + 486662x^2 + x$ where k is the prime field GF($2^{255} - 19$)

Scalar multiplication is an essential building block for public-key elliptic curve cryptography and has a significant influence on the execution time of ECC algorithms. Therefore, optimized scalar multiplication methods are vital for good ECC performance.

2 Outline

The goal of this project is to optimize an elliptic curve-based scalar multiplication algorithm using an appropriate parallelization strategy. The project will begin by conducting a literature review of the existing algorithms [1] and selecting an elliptic curve and scalar multiplication algorithm for parallelization. In the following weeks, the project will analyze the selected algorithm, identify its parallelizable components, and design a parallelization strategy based on the identified components. The implementation of the parallelized algorithm will be carried out using Python (numba for speeding up). The implementation will be automated over various elliptic curves to test its scalability. The project will identify potential performance bottlenecks and explore ways to mitigate them. Parallel hardware, such as GPUs or multi-core CPUs, will be used for testing and optimization of the algorithm, with necessary tools and libraries, such as OpenMP or CUDA (via Compyle), used for parallelization. The parallelized algorithm's performance will be evaluated and compared with the original sequential algorithms, and the project's findings and conclusions will be documented in a final report. Finally, the project will discuss any limitations or future research directions for improving the algorithm.

3 Deliverables

The end results would be python or scripts showcasing the implementations of various algorithms for speeding up scalar multiplication over various standard elliptic curves. The scripts will be automated to test over all the curves under consideration, and optimized to be best of our abilities, the final result will be a plot visualizing the times taken across various curves over various implementations.

4 Timeline

There are a total of 6 weeks available. This is a rough timeline which may change over the course of the project to accommodate for any hurdles faced in the previous weeks. There may be changes to the work itself depending on the direction we take as the course proceeds.

- Week 1: Research and Familiarization
  - Familiarization with the elliptic curve cryptography and scalar multiplication algorithms.
  - Research of existing parallelization techniques and related works.
  - Selection of an appropriate elliptic curve and scalar multiplication algorithm for parallelization.
- Week 2: Algorithmic Analysis and Design
  - Analysis of the selected algorithm and identification its parallelizable components.
  - Identifying potential performance bottlenecks and explore ways to mitigate them.
- Week 3: Code Implementation and Serial Optimization
  - Implementation of the algorithm in Python (using libraries like pycrytodome, tinyec as reference).
  - Conducting initial performance tests and identify any areas for optimization.
  - Optimization using techniques such as loop unrolling, vectorization, and cache optimizations.

- Week 4: Parallel Hardware and Tools
  - Identifying the parallel hardware that will be used for testing and optimizing the algorithm.
  - Using necessary tools and libraries, such as OpenMP or CUDA (via Compyle), to optimize and parallelize the code.
  - Testing the parallelized code on the selected hardware and analyze the results.
- Week 5: Performance Evaluation and Comparison
  - Evaluating the performance of the parallelized algorithm and comparing it with the original sequential algorithm.
  - Automating over various elliptic curves (Automan)
  - Documenting the results and identify any performance gains or losses.
  - Exploring ways to further optimize the parallelized algorithm, if necessary.
- Week 6: Conclusion and Future Work
  - Summarizing the findings and conclusions regarding the parallelization of elliptic curve-based scalar multiplication.
  - Discussing any limitations or future research directions for improving the algorithm.
  - Writing a final report documenting the research, design, implementation, and evaluation of the parallelized algorithm.

5 Git repository

The Git repository that will be used for version control is linked here. The repository is kept public for view access to the instructor and TAs.

References

[1] Gerwin Gsenger. "Speeding Up Elliptic Curve Cryptography by Optimizing Scalar Multiplication in Software Implementations". MA thesis. Institute for Applied Information Processing and Communications (IAIK) Graz University of Technology, 2014.