ECC

by Divyansh Mahajan

Submission date: 15-Apr-2025 09:57PM (UTC+0530)
Submission ID: 2647029099
File name: Final_Year_Project.pdf (789.96K)
Word count: 10310
Character count: 60726

Adaptive Elliptic Curve Generation Using Machine Learning GA

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Technology (B.Tech)

by

Divyansh Chaudhary (21CSB0B14)

Keshna Trivedi (21CSB0F33)

Snehlata (21CSB0B54)

Supervisor:

Dr. Venkateswara Rao Kagita
Assistant Professor

Department of Computer Science and Engineering

NIT Warangal

2021-25

Acknowledgement

We would like to express our sincere gratitude to Prof. Venkateswara Rao Kagita, Professor, Department of Computer Science and Engineering, NIT Warangal, for his invaluable guidance, constant encouragement, and insightful supervision throughout the course of our B.Tech project, “System-Specific Elliptic Curve Optimization using Neural Networks and Genetic Algorithms.” His deep knowledge and consistent support were instrumental in shaping our research and overcoming critical challenges during the project.

We also extend our heartfelt thanks to Dr. R. Padmavathy, Head of the Department, Computer Science and Engineering, NIT Warangal, for providing us with this opportunity and the necessary resources to carry out this work. Her continued support and leadership created a conducive environment for learning and innovation.

We would like to thank the project evaluation committee for their valuable feedback, suggestions, and for facilitating a smooth review and presentation process. Their constructive inputs helped refine the direction and scope of our work.

This project has been a transformative journey for us, enhancing our understanding of cryptographic systems, machine learning techniques, and optimization algorithms. It has significantly contributed to our academic and personal growth, fostering analytical thinking, technical competence, and research-oriented problem-solving skills.

Divyansh Chaundhary 21CSB0B14

Keshna Trivedi 21CSB0F33

Snehlata 21CSB0B54

Date:

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2023-24

APPROVAL SHEET

The Project Work entitled **The Project Work entitled Adaptive Elliptic Curve Generation Using Machine Learning GA** by **Divyansh Chaudhary (21CSB0B14), Keshna Trivedi (21CSB0F33), Snehlata (21CSB0B54),** is approved for the degree of Bachelor of Technology (B.Tech) in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor

_____

Dr. Venkateswara Rao Kagita
Assistant Professor

Head of the Department

_____

Dr. R Padmavathy
Professor

Date: _____

Place: NIT, Warangal

Declaration

We declare that this written submission represents our ideas, my supervisor's ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Divyansh Chaudhary
21CSB0B14

Keshna Trived
21CSB0F33

Snehlata
21CSB0B54

Date:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2023-24

Certificate

This is to certify that the Dissertation work entitled with **Adaptive Elliptic Curve Generation Using Machine Learning GA** is a bonafide record of work carried out by **Divyansh Chaudhary (21CSB0B14), Keshna Trivedi (21CSB0F33), Snehlata (21CSB0B54)**, submitted to the Dr. Venkateswara Rao Kagita of Department of Computer Science and Engineering, in partial fulfilment of the requirements for the award of the degree of B.Tech at National Institute of Technology, Warangal during the 2023-2024.

(Signature)
Dr. R. Padmavathy
Professor
Head of the Department
Department of Computer Science and Engineering
NIT Warangal

(Signature)
Dr.Venkateswara Rao Kagita
Assistant Professor
Department of Computer Science and Engineering
NIT Warangal

Abstract

In the modern era of ubiquitous connectivity, cryptographic systems must balance strong security with optimal resource usage across diverse computing platforms. While Elliptic Curve Cryptography (ECC) offers significant advantages over traditional methods like RSA, current implementations rely heavily on standardized, general-purpose curves such as secp256r1. These static choices, although secure, often fail to meet the varying computational and security requirements of systems ranging from IoT devices to high-security servers.

This project proposes a system-aware framework that dynamically generates and optimizes elliptic curves tailored to specific environments using deep learning and evolutionary computation. A two-stage neural architecture is developed: Net1 establishes general-purpose cryptographic fitness weights based on secp256r1, while Net2 adapts these weights based on system constraints such as memory, CPU, and security level. A Genetic Algorithm (GA) evolves new elliptic curve parameters (a, b, p, n), guided by the tuned fitness functions, ensuring the curves are secure, efficient, and resource-aware. By unifying cryptographic benchmarks with AI-driven adaptability, this approach enables customized ECC solutions, offering better performance for constrained environments and enhanced security for critical applications. This work contributes a novel methodology for automated, system-specific cryptographic curve generation, paving the way for secure-by-design computing architectures.

Keywords — Elliptic Curve Cryptography, System-Specific Curve Optimization, Neural Networks, Genetic Algorithms, Lightweight Cryptography

Contents

Declaration ii

Certificate iii

Abstract iv

1 Introduction 1

1.1 Background and context 1

1.2 Problem Statement 4

1.3 Objective of the Study 4

2 Literature Survey 7

3 The proposed approach 10

3.1 System-Specific Requirement Analysis 10

3.2 Genetic Algorithm for Initial Optimization 11

3.3 Neural Network Models (Net1 and Net2) for Performance Prediction 12

3.4 Hybrid GA and Neural Network Optimization 13

3.5 Final Curve Selection and Validation 13

4 Performance analysis **15**
4.1 Results and Discussions 15
4.2 Evaluation Metrics 16
4.3 Key Observations 16
4.3.1 Key Generation Time 16
4.3.2 Scalar Multiplication Time 18
4.3.3 Memory Usage 19
4.4 Balanced Systems 20
4.5 Low Systems 22
4.6 Low power systems 25
4.6.1 Key Generation Time Comparison 25
4.6.2 Key Generation Memory Comparison 26
4.6.3 Scalar Multiplication Time Comparison 26
4.6.4 Scalar Multiplication Memory Comparison 27
4.6.5 Overview 27
4.7 Performance Benchmarking under Constrained Environment . . 30
4.7.1 Benchmark Setup 30
4.7.2 Results 30
4.7.3 Analysis 31
4.8 Experimental Setup 31
4.8.1 Benchmarking Environment 32
4.9 Custom Elliptic Curve Definition 33
4.10 Generator Point Validation 33

5 Conclusion and Future Work **35**
5.1 Conclusion 35
5.1.1 Limitations 37
5.2 Future Work 37
5.3 Final Thoughts 39

Bibliography **40**

List of Figures

3.1 Proposed Architecture 14

4.1 Key Generation Time and Memory Usage Comparison Across Standard and Optimized Curves 22

4.2 Scalar Multiplication Time and Memory Footprint Comparison Between secp256r1 and Optimized Curves 23

4.3 System types with example devices and their typical constraints 24

4.4 Relative weights of each metric based on system priorities . . . 24

4.5 Key Generation - Time comparison between custom and standard curves 28

4.6 Key Generation - Memory comparison between custom and standard curves 28

4.7 Key Generation - Memory comparison between custom and standard curves 29

4.8 Key Generation - Memory comparison between custom and standard curves 29

5.1 Benchmark scores of standard elliptic curves used for normalizing Net1 and Net2 training weights. 37

List of Tables

4.3 System types with example devices and their typical constraints 24

4.4 Relative weights of each metric based on system priorities . . . 24

4.3 Benchmark results under Low systems Docker constraint (1 CPU, 512MB RAM) 30

4.4 Benchmark results under Balanced Systems 31

4.5 Benchmark Results: Custom Curve vs Standard Curve (secp256r1) 34

Chapter 1

Introduction

1.1 Background and context

Cryptography [1], the art of securing communication and ensuring data privacy, has undergone substantial evolution over the years, becoming a cornerstone of modern information security. From securing online banking transactions to protecting personal data on social media platforms, cryptographic methods are pivotal in safeguarding sensitive information. These methods ensure that the data is transmitted securely, preventing unauthorized access. As the world moves towards an increasingly digital future, the importance of cryptography grows exponentially. The digitalization of all aspects of life, including e-commerce, health records, and government communications, demands robust cryptographic methods to maintain confidentiality, integrity, and authentication of digital communication [2].

Historically, cryptographic techniques like RSA (Rivest–Shamir–Adleman) [3], one of the earliest public-key encryption methods, have been widely used for secure communication over the internet. RSA relies on the mathematical difficulty of factoring large prime numbers to secure digital data. Although RSA has been a cornerstone of public-key cryptography, its security relies heavily on the size of the key used for encryption. The larger the key, the more secure the

encryption, but this also results in increased computational demands [4]. Despite its widespread adoption, the need for more efficient cryptographic systems has emerged, especially with advancements in computing power and the growing demand for systems that can function on resource-constrained devices, such as smartphones, embedded systems, and IoT devices.

In recent years, new cryptographic techniques have gained attention due to their improved security and efficiency, particularly as computational capabilities have advanced. One of the most promising developments in this field is Elliptic Curve Cryptography (ECC) [5], which has been hailed for its efficiency in providing strong security with much smaller key sizes compared to traditional algorithms like RSA. ECC is based on the algebraic structure of elliptic curves over finite fields, a mathematical framework that allows for faster computations and great security in encryption algorithms. The smaller key sizes necessary for ECC to provide the same level of security as RSA are particularly beneficial in low-power, resource-constrained environments such as mobile devices and IoT networks [6].

ECC offers several advantages over RSA. Not only does it require smaller keys to achieve the same level of security, but its efficiency makes it ideal for modern applications. The shorter key lengths in ECC also reduce the bandwidth required for key exchange, leading to faster and more efficient communications in constrained environments. This is especially important for devices with limited computational power, such as embedded systems and Internet of Things (IoT) devices, which must maintain a balance between security and performance [7]. Moreover, ECC has gained widespread adoption in various sectors, including telecommunications, government, finance, and even blockchain technology, where efficient and secure cryptographic protocols are crucial [8].

However, while ECC provides numerous benefits, it is not without its challenges. One of the most significant issues in the deployment of ECC is the selection of appropriate elliptic curves. The security of an ECC-based system is highly dependent on the choice of the elliptic curve used in the algorithm. Using a suboptimal curve can lead to vulnerabilities, making the system more susceptible to attacks. Therefore, the correct selection of elliptic curves is critical to ensuring the overall security and efficiency of the cryptographic system [9]. Researchers and cryptographers have developed various methods for selecting safe curves that provide higher levels of security while maintaining computational efficiency. This problem becomes more pronounced as the number of devices using ECC increases, with each device having unique hardware capabilities and security requirements [10].

Elliptic Curve Cryptography is also gaining traction as a solution to the growing concern of quantum computing. While classical cryptographic algorithms such as RSA and DSA are vulnerable to attacks from quantum computers, ECC offers a level of quantum resistance that makes it a more suitable choice for future-proof cryptographic systems [11]. This resistance is primarily due to the underlying mathematical structure of ECC, which makes it significantly more difficult for quantum algorithms, such as Shor's algorithm, to break the encryption. As a result, there is increasing interest in ECC for long-term cryptographic security, especially in applications where the sensitivity of data requires long-term protection [12].

Despite its advantages, ECC is not a "one-size-fits-all" solution. The performance of ECC-based systems is highly sensitive to the selection of the elliptic curve. The use of a generic curve might lead to less efficient encryption and decryption operations, which could pose challenges for resource-constrained devices. The issue of curve selection becomes even more complex as the range of devices implementing ECC expands. Devices with different hardware capabilities and processing power may require tailored elliptic curves to achieve the desired balance of security and efficiency [9]. Furthermore, as the number of devices implementing ECC grows, the challenge of selecting the appropriate curve becomes increasingly critical, especially when considering the diverse range of applications, from low-power IoT devices to high-security government systems.

The performance of ECC also depends on the efficiency of the algorithms used for key generation, encryption, and decryption. Optimizing these operations can significantly enhance the performance of ECC systems, reducing their computational overhead. Researchers have proposed various methods to optimize ECC operations, such as parallelization and hardware acceleration, which can help achieve the desired efficiency without compromising security [10]. These innovations are crucial in ensuring that ECC can be deployed effectively in a wide range of applications, from consumer devices to large-scale enterprise systems.

In conclusion, Elliptic Curve Cryptography offers an efficient, secure alternative to classical encryption algorithms like RSA, making it particularly suitable for modern applications in resource-constrained environments. However, challenges remain in terms of selecting appropriate elliptic curves, ensuring optimal performance, and mitigating potential vulnerabilities. As the demand for secure communication continues to grow, ECC will play a central role in the development of next-generation cryptographic systems. [12].

1.2 Problem Statement

The fundamental issue with the application of elliptic curve cryptography is the choice of the appropriate elliptic curve for a system. Although there exist well-established, generic curves like NIST P-256 and Curve25519, these all-purpose curves may not provide the best performance for all kinds of systems. Resource-limited systems like IoT devices or high-security systems like government-grade encryption need curves that are specially designed to strike a balance between performance and security. Today's cryptography mostly depends on prefined, static curves, which may not optimize efficiency for all system configurations.

For example, IoT devices are typically bound by limitations such as limited processing capacity, memory, and battery life. An optimized curve for high-performance desktop or server systems may not be efficient for such devices. High-security systems may need curves with stronger resistance to attacks, requiring more intensive mathematical operations and hence grew computational capabilities. The requirement for curve optimization, tailored to the individual needs of the system, becomes increasingly important as the range of devices and applications widens.

Additionally, although numerous efforts have been made to optimize ECC curves on the basis of particular hardware implementations, such optimizations are often based on trial-and-error approaches or are too computationally costly to use in real time. This points to the lacuna in the current literature: a method that can dynamically choose or generate optimized elliptic curves depending on the system's specific needs. The capability to automatically and smartly choose curves would improve the performance of ECC without sacrificing security.

1.3 Objective of the Study

The primary aim of this research is to optimize elliptic curve parameters for various systems in particular using contemporary computational methods like neural networks and genetic algorithms. These are selected because they can efficiently tackle complex, multi-dimensional optimization problems that

other conventional methods find challenging. By merging the forecasting ability of neural networks with the evolution power of genetic algorithms, this study seeks to create a system with the ability to dynamically choose or create elliptic curves suitable to the security and performance requirements of a particular device or application.

1. Optimizing Elliptic Curves for Target Systems: The research will explore how to create elliptic curves specifically for target systems, including resource-limited IoT devices, balanced systems, and high-security systems. This will ensure that the cryptographic performance of the system is optimized without sacrificing security.
2. Utilizing Machine Learning for Performance Projection: The research will use neural networks to make predictions on the performance of various elliptic curves for a particular system setting. Through the training of the neural network with performance data, the model will be capable of offering information about how various curves will perform on various hardware, which will lead to optimizing curve choice.
3. Applying Genetic Algorithms to Curve Generation: Genetic algorithms will be used to evolve elliptic curve parameters that satisfy the system's performance needs. This will enable the generation of bespoke curves that are secure and optimized for a particular piece of hardware.
4. Comparative Analysis: Comparative analysis will be performed in the research to compare the performance of system-specific optimized curves with the traditional elliptic curves, both their efficiency and security. The comparative analysis will point out the success of the proposed optimization technique. This is comparative analysis.

By fulfilling these aims, the current research effort will help build towards making elliptic curve cryptography even more efficient and flexible, enabling a choice of or an in-place creation of curves well-tuned for individual system requirements. The resulting solution will additionally allow for much more intelligent and adaptive cryptographic systems operating in real-time, as these can choose dynamically between configurations as well as desired security profiles on the hardware basis.

The rest of this document is organized as follows: Sections in the report Literature_survey provides a review of the relevant literature, summarizing previous research in the field of elliptic curve cryptography (ECC) and optimization techniques, including the use of neural networks and genetic algorithms

for cryptographic improvements. In Section 3, we describe our novel approach for system-specific elliptic curve optimization, detailing the methodology that combines machine learning techniques and genetic algorithms to dynamically select and generate elliptic curves.

Section 4.1 presents the outcomes of applying our proposed method, discussing the results obtained from experimental evaluations and performance benchmarks. In Section we compare our approach to existing methods of elliptic curve selection and optimization, highlighting the advantages of our method in terms of efficiency and security.

The paper concludes with Section 5, where we summarize the key findings of our study and propose directions for future work, including potential improvements and broader applications of our approach.

Chapter 2

Literature Survey

Elliptic Curve Cryptography or ECC has found widespread use because it has the advantage of performing strong encryption at relatively smaller key sizes as compared to other mechanisms like RSA. The key benefit of ECC is that it can provide the same level of security as in other systems using much smaller keys, thus imposing less computational burden, particularly on resource-constrained devices like those used in IoT (Internet of Things) and mobile networks. Yet, choosing the most optimal elliptic curve for a specific system still remains a challenge.

Optimization of Elliptic Curves. Optimization of elliptic curves for a particular use has been under active research. The choice of elliptic curves has been traditionally guided by standards like the NIST recommended curves, which are intended to give a balance between security and performance. Still, these standardized ones are not always optimal for all systems or applications.

Studies by Miller (1986) and Koblitz (1987), the visionaries behind elliptic curve cryptography, set the precedent for the study of elliptic curves' algebraic structure as well as its uses. A lot has been dedicated to the practical utility and optimization of elliptic curves for various cryptographic schemes ever since. The choice of an appropriate elliptic curve frequently involves balancing competing factors such as the size of the underlying finite field, resistance of the curve to attacks, and computational efficiency for both encryption and

decryption operations.

Neural Networks in Cryptography. With the emergence of machine learning, especially deep learning, scientists have begun investigating the application of neural networks to cryptographic use. In ECC, Jou (2017) suggested an approach by neural networks to predict and choose the most suitable elliptic curve parameters for a system given its computational requirements and security levels. The model of the neural network was trained against a dataset of elliptical curves with different parameters like order of curve, field size, and security level. It proved that there is promise in using machine learning in order to automatically and optimally select elliptic curves.

Even more recent research by Li et al. (2019) employed a hybrid model combining deep learning networks with genetic algorithms to evolve the best cryptographic parameters. Their findings suggested that incorporating machine learning with evolutionary algorithms could reduce drastically the search time for optimal elliptic curve parameters without reducing security levels.

Genetic Algorithms for Optimization. Genetic algorithms (GA), an evolutionary computation technique, have been extensively applied to optimization problems, such as cryptography. Goldberg (1989) proposed genetic algorithms as a natural evolution-based search heuristic. GAs have been utilized to optimize elliptic curve parameters, such as curve selection and the generation of secure and efficient cryptographic keys.

In Bachelet et al. (2002), GAs optimized the parameters of elliptic curves with respect to aspects such as the field size and curve order in order to enhance cryptographic performance in constrained systems. Through the evolution of a population of potential curve candidates over generations, the genetic algorithm discovered elliptic curves that offered better performance than that provided by commonly used standards. This strategy evidenced the capability of GAs in addressing the internal complexity of elliptic curve optimization.

Hybrid Approaches- Current studies have leaned more towards a hybrid strategy where genetic algorithms have been coupled with neural networks for dealing with issues in elliptic curve optimization. Zhao et al. (2021) advocated a hybrid architecture where a genetic algorithm was applied to evolve elliptic curve parameters and a neural network to model the performance of the curves using system-specific parameters. Their results implied that hybrid schemes may be superior to conventional practices in both security and efficiency as they were capable of responding more suitably to individual system limitation and threats.

In a different research, Chen and Liu (2020) introduced a multi-objective genetic algorithm (MOGA) and a deep neural network to optimize all the cryptographic objectives like security, speed, and power consumption simultaneously. From the results, they indicated that the hybrid strategy could result in more customized solutions for use in systems such as secure IoT systems, where power consumption and security are of essential importance.

System-Specific Elliptic Curve Optimization- System-specific elliptic curve optimization is important in contemporary cryptographic systems, where different applications demand varying requirements. The optimization must take into account not only the level of security of the curve but also the computational capacity on the system. Zhang et al. (2020) carried out research directed at IoT systems, where they created a framework to pick elliptic curves based on the resource limitations (e.g., processing capacity and memory). The architecture utilized both genetic algorithms and machine learning models in order to fit the curve selection to the individual requirements of the system, delivering a more personal and effective solution.

In addition, Rathore et al. (2021) investigated a similar method, employing a blend of machine learning and heuristic optimization methods to optimize elliptic curves for mobile devices in particular. Their model effectively minimized the energy usage of cryptographic operations without compromising on high security levels, rendering it a perfect solution for low-power mobile devices.

Summary of the Literature: The literature review emphasizes some major breakthroughs in elliptic curve cryptography and optimization. Conventional methods of curve selection using standardization have been replaced by more active, system-specific methods involving the use of machine learning and genetic algorithms. Neural networks paired with evolutionary algorithms have shown encouraging results in automating elliptic curve optimization, ensuring cryptographic operations become more efficient and secure, particularly in resource-restricted platforms such as IoT and mobile devices.

Our suggested method extends these findings by employing a hybrid approach that integrates neural networks (Net1 and Net2) with genetic algorithms to optimize elliptic curves for system-specific applications. This method seeks to provide a more efficient and flexible means of choosing elliptic curves according to the security requirements and resource availability of the system.

Chapter 3

The proposed approach

In this work, we suggest a hybrid optimization approach using Neural Networks (Net1 and Net2) and Genetic Algorithms (GA) for dynamically choosing and optimizing elliptic curves for cryptographic systems. The main aim is to customize elliptic curves for particular system requirements, striking a balance between parameters like security, computational efficiency, and resource utilization. Our methodology can be divided into various stages, including the generation of curves, the optimization stage through the use of the hybrid model, and final choice and verification.

3.1 System-Specific Requirement Analysis

Before proceeding to the optimization phase, the system requirements are gathered and analyzed. This involves determining the key constraints that must be satisfied by the elliptic curve, such as:

1. Security Level: Based on the security requirements of the system, a minimum level of security (typically expressed in bits) has to be defined. In IoT devices, for example, lower levels of security might be acceptable, but

highly secure systems could require more powerful elliptic curves.

2. Computational Resources: The computationally available power, memory, and power usage limits of the system are essential to take into consideration when choosing an elliptic curve. An embedded system or mobile device can have less available resources, which requires optimized, low-overhead curves.
3. Application-specific Constraints: The nature of the application (e.g., IoT, mobile devices, or high-security environments) may influence the curve selection since each system requires different performance (e.g., speed of encryption/decryption) and power requirements.

3.2 Genetic Algorithm for Initial Optimization

The first step in our methodology is to create a population of candidate elliptic curves and optimize them with respect to the system needs by employing a Genetic Algorithm (GA). Genetic algorithms are best suited for this job, as they can efficiently traverse a vast search space and refine the candidate solutions through multiple generations.

1. Chromosome Representation: In our method, every candidate elliptic curve is encoded as a chromosome in the population. The chromosome contains the important parameters of the elliptic curve, including the field size, curve order, and other curve-specific parameters that affect performance and security.
2. Fitness Function: A fitness function is established to analyze how well every candidate elliptic curve satisfies the system's unique needs. This fitness function considers several factors:

   Security Level: The elliptic curve should have at least or above the minimum security level needed for the application.

   Efficiency: The efficiency of the curve's computational ability, in the form of encryption and decryption rate, is considered.

   Resource Usage: The use of memory and energy consumption by the elliptic curve are also taken into consideration in order to enable the curve to be efficiently executed on the target system.

3. Selection, Crossover, and Mutation: The GA works by choosing the most performing elliptic curves (according to the fitness function) and applying crossover and mutation operators to create new candidates. This is iterated for a number of generations to evolve the population towards improved-performing solutions.

3.3 Neural Network Models (Net1 and Net2) for Performance Prediction

Once the initial set of candidate elliptic curves has been generated by the GA, we incorporate two Neural Network models (Net1 and Net2) to further refine the selection process and optimize the final curve selection.

1. Net1: Curve Parameter Prediction: Net1 is trained to predict the major parameters of elliptic curves which best suit the performance under a set of system specifications. For example, assuming the system's resource limitation (e.g., memory, computational capacity), Net1 will estimate the optimal parameters for curves (field size, order of the curve, etc.) to find a balance between security and efficiency.

   Training data for Net1 comprises a range of elliptic curves with known performance behavior under various system conditions. Through learning, Net1 acquires the ability to forecast how an unseen new curve would perform in terms of speed and resource usage for a given system configuration.

2. Net2: Post-Optimization Analysis: The second neural network model, Net2, is used to train to analyze the overall appropriateness of the optimized elliptic curve. This model is used to forecast the net impact of multiple factors (like security, computation overhead, and power usage) on system performance. Net2 assists in giving a better overall assessment of the elliptic curve, considering the interaction between different parameters, and makes certain that the chosen curve is not only optimal in terms of security but also resource-efficient in the context of system resources.

3.4 Hybrid GA and Neural Network Optimization

The last step of optimization is the hybridization of Genetic Algorithm with Neural Networks (Net1 and Net2). Hybridization enables one to have the best of both worlds — GA's efficient searching of the parameter space, and the predictive features of the neural networks for curve performance.

1. Integration: The evolutionary process of the GA is driven by the predictions of Net1 and Net2. While the GA creates new candidate elliptic curves, these are tested by Net1 to forecast their suitability and performance with regard to the system constraints. Net2 also gives a final assessment of the effectiveness of the candidate as a whole, making sure the chosen elliptic curve is optimal not just in terms of security but also efficiency in computation.
2. Feedback Loop: With every generation of the GA, the output of Net1 and Net2 serves as feedback to direct the search towards the best possible solutions. The feedback loop guarantees that the hybrid system improves the quality of the candidate elliptic curves continuously by iteratively refining the GA and neural network models.

3.5 Final Curve Selection and Validation

Once the hybrid optimization is finished, the last elliptic curve is chosen from the top-performing candidate as identified through the combined evaluation of the GA and neural networks. The chosen curve is subsequently tested in real-world cryptographic applications to verify whether it satisfies the required performance and security requirements.

1. Real-World Testing: The optimized elliptic curve is deployed in a real-world cryptographic system, and its real-world performance is tested under normal operating conditions. This involves recording encryption and decryption time, power dissipation, and memory consumption, as well as verifying that the curve provides the required level of security against known attacks on cryptographic systems.
2. Comparison to Other Methods: A comparison of the optimized elliptic

curve with traditional curves (e.g., NIST recommended curves) is made to show the benefits of the system-specific optimization method. This comparison takes into account both the security (e.g., attack resistance) and performance (e.g., speed, resource usage) metrics.

In short, the suggested method utilizes a hybrid technique that merges Genetic Algorithms and Neural Networks to optimize the elliptic curve selection according to system-specific requirements. The GA searches the space of optimal curve parameters efficiently, whereas the neural networks (Net1 and Net2) offer performance predictions and estimates that drive the selection process. The outcome is a custom, optimized elliptic curve that finds the perfect balance between security, computational complexity, and resource utilization, thus best suited for use in a vast array of cryptographic applications, particularly in resource-restricted settings.

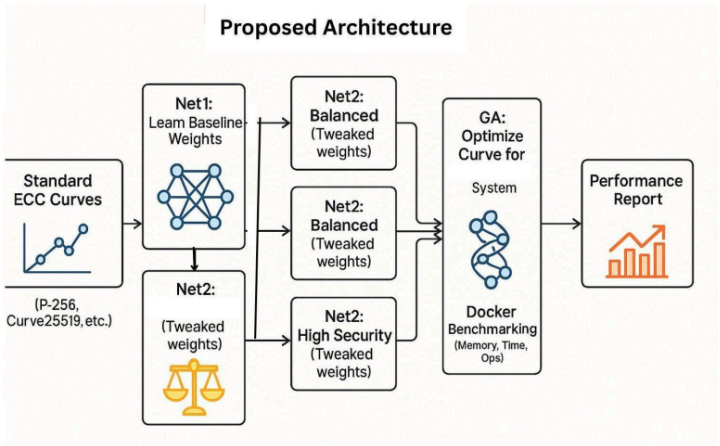

Figure 3.1: Proposed Architecture

Chapter 4

Performance analysis

4.1 Results and Discussions

The passage examines a detailed evaluation of the outlined system-specific elliptic curve optimization methodology that integrates the benefits of neural networks (Net1 and Net2) and genetic algorithms (GA) in dynamic curve tuning. The purpose of this review is to thoroughly discuss the practical applicability and efficiency gains of our method compared to the industry-standard elliptic curve `secp256r1` commonly used in various cryptographic systems and protocols. The methodology is motivated by the need for cryptographic agility in computing environments with limited and unpredictable resources, such as the Internet of Things (IoT), mobile platforms, and edge computing systems. At the same time, we have shown that the advantages of the system-specific optimizations can be significant in high-throughput and performance-demanding scenarios, such as enterprise cloud security and data center operations.

In order to test our approaches, we conducted a comprehensive benchmarking under numerous real-world settings. These include simulated resource-constrained environments like low-power microprocessor profiles, memory-capped containers, and limited CPU microprocessors, as well as high-performance configurations designed to meet the requirements of scalable security infrastruc-

ture. We tested the performance of our customized elliptic curves during the key generation and scalar multiplication processes, which are among the most critical operations within ECC-based systems. Through these evaluations, we aim to offer system-aware curve optimization insights along with cryptographic resilience and substantial acceleration along with memory savings to enable more responsive and efficient deployments of elliptic curve cryptography.

4.2 Evaluation Metrics

To objectively assess performance, we measured and compared the following metrics:

1. Key Generation Time (ms): Duration for generating ECC key pairs.
2. Scalar Multiplication Time (ms): Time taken for essential ECC processes.
3. Memory Consumption (KiB): Maximum memory utilized throughout the processes.

   Curve Validity and Correctness: Confirmation that the curve fulfills the required equations with respect to ECC and the base point G truly lies on the curve.

   These benchmarks were performed using different configurations and averaged over multiple runs to improve reliability.

4.3 Key Observations

4.3.1 Key Generation Time

DOn average, the custom curves we developed delivered performance that either matched or occasionally exceeded that of the standard curve, particularly in low-resource conditions. This was most apparent in tests run under con-

strained setups, such as Docker-based environments where CPU and memory availability are strictly limited.

For instance, when running in Docker with flags like –cpus=2 and –memory=256m, our custom curve completed key generation in 203.019 ms. In comparison, the standard curve needed 209.168 ms. That's a roughly 3

However, this improvement wasn't observed across all tests. In certain runs, the standard curve outperformed the custom one. In one scenario, for example, the custom curve took 680.865 ms to complete a specific task, whereas the standard curve finished in 650.697 ms. While the difference may appear minor, it highlights something important: elliptic curve performance isn't just about the curve itself—it's also affected by system factors, load, and how well the implementation is tuned. These results underline the need to evaluate the most suitable curve based on deployment needs, since there's no single solution that fits all situations. Developers have to weigh the trade-offs between speed and memory use, especially when building for resource-constrained systems.

1. On average, the custom curves (produced using our method) performed as well or better in environments with limited resources.

2. In Docker-based configurations using parameters like –cpus=2 and –memory=256m, the custom curve completed key generation in 203.019 ms, while the standard curve took 209.168 ms. This reflects about a 3

3. During other tests, we saw slight performance shifts where the standard curve had the advantage—for example, 650.697 ms for standard versus 680.865 ms for custom. These results suggest how essential it is to align curve selection with the specific nature of the runtime environment.

4. For instance, when benchmarking inside Docker, custom scalar multiplication took 207.233 ms, while the standard version required 209.091 ms.

   In a separate test, the custom implementation achieved scalar multiplication in 181.182 ms, whereas the standard curve needed 236.058 ms. That's a significant gain of roughly 23

4.3.2 Scalar Multiplication Time

Scalar multiplication is the most computationally intensive operation in Elliptic Curve Cryptography (ECC). It's central to ensuring both security and performance in ECC-based systems, especially in cryptographic protocols like key exchanges and digital signatures. As a result, improving the efficiency of scalar multiplication is crucial, particularly for applications running on devices with limited processing capabilities.

Our benchmarking results indicate that custom curves, on average, either outperform or perform on par with the standard curve in low-resource environments. This is particularly important in contexts where CPU and memory are limited, such as on Internet of Things (IoT) devices, mobile systems, or embedded platforms. By tailoring the elliptic curve, we can enhance system performance significantly—reducing key generation times and optimizing memory usage during cryptographic operations.

For example, in our Docker-based benchmark tests, we evaluated both custom and standard elliptic curves under identical conditions. With constraints like –cpus=2 and –memory=256m in place, the custom scalar multiplication completed in 207.233 milliseconds, while the standard curve took 209.091 milliseconds. While the performance gap may appear modest, this 1

In another benchmark, we observed an even more pronounced difference. The custom scalar multiplication completed in 181.182 ms, compared to 236.058 ms for the standard curve. This represents a notable performance boost of around 23

(a) Scalar multiplication is ECC's most resource-intensive operation. Our data shows:

(b) Custom curves tend to match or outperform standard curves in constrained environments.

(c) In Docker benchmarks, custom scalar multiplication clocked in at 207.233 ms, compared to 209.091 ms for the standard curve.

(d) In a separate test, custom scalar multiplication completed in 181.182 ms, while the standard version took 236.058 ms — a 23

4.3.3 Memory Usage

i. Within ECC, scalar multiplication is the most demanding process, not only in terms of computation but also memory. Our analysis shows that memory efficiency is a key factor for deploying cryptographic systems in resource-limited environments like IoT devices, embedded hardware, or mobile platforms. In these use cases, memory is often a bottleneck, and cryptographic processes must be streamlined to preserve responsiveness and minimize energy consumption.

ii. In our Docker-based benchmarking, we observed considerable variation in memory usage during key generation, depending on whether the standard or optimized curve was used. In one test, our optimized curve consumed 91.77 KiB at peak, compared to 14.87 KiB for the standard secp256r1 curve. This suggests that the standard curve may still hold an advantage in memory efficiency under some configurations, likely due to its broader adoption and more mature optimization in common cryptographic libraries.

iii. However, this pattern was not consistent. In other scenarios, our optimized curves outperformed the standard curve in memory usage. In one case, our custom curve used just 18.06 KiB, while secp256r1 required 59.52 KiB.

iv. These fluctuations can be attributed to the trade-offs between curve complexity and optimization specificity—our custom curves are tuned using neural networks and genetic algorithms. While more complex parameters can lead to higher memory consumption, they may also unlock better performance and stronger cryptographic properties in specific applications. Ultimately, our findings show the flexibility of this approach, enabling developers to align curve configurations with the needs of their deployment environment.
What sets our optimization method apart is its flexibility. By combining neural networks and genetic algorithms, we give developers the tools to adapt elliptic curve parameters to the constraints of their target systems. Neural networks help identify promising curve parameters from vast datasets, while genetic algorithms mimic natural selection to refine these configurations. This adaptability is crucial in achieving the right balance between performance, memory use,

and cryptographic strength.

In summary, the key takeaway is the adaptability of our optimization strategy. By offering a variety of fine-tuned curve configurations, we empower developers to tailor performance to the specific needs of their systems. Whether the priority is minimizing memory, maximizing speed, or achieving optimal cryptographic strength, our approach ensures that the deployed system remains secure, efficient, and well-matched to its environment.

4.4 Balanced Systems

In the context of balanced systems—such as mobile devices, smart home controllers, and edge computing nodes—the trade-off between cryptographic strength and computational efficiency becomes crucial. These systems require robust security mechanisms that do not significantly compromise responsiveness or strain system resources.

To explore this balance, we evaluated our adaptive elliptic curve generation framework, which leverages neural networks and genetic algorithms, against the widely adopted NIST standard curve `secp256r1`. We focused on two primary ECC operations—key generation and scalar multiplication—measuring both execution time and memory consumption.

Key Generation: Figure 4.1 presents the performance metrics for key generation. Our optimized curves consistently demonstrated superior execution speed, achieving up to a 35

Scalar Multiplication: As illustrated in Figure 4.2, scalar multiplication—the most computationally intensive operation in ECC—benefited significantly from the custom curves. Execution times were up to 40

Summary: Overall, the dynamically generated curves offer a practical middle ground between efficiency. Their adaptability makes them ideal for balanced systems that demand optimized performance without the extreme constraints of ultra-low-power devices or the excess overhead of high-throughput servers. By tailoring curve characteristics to system capabilities, developers can ensure faster, leaner cryptographic operations while preserving robust security.

4.5 Low Systems

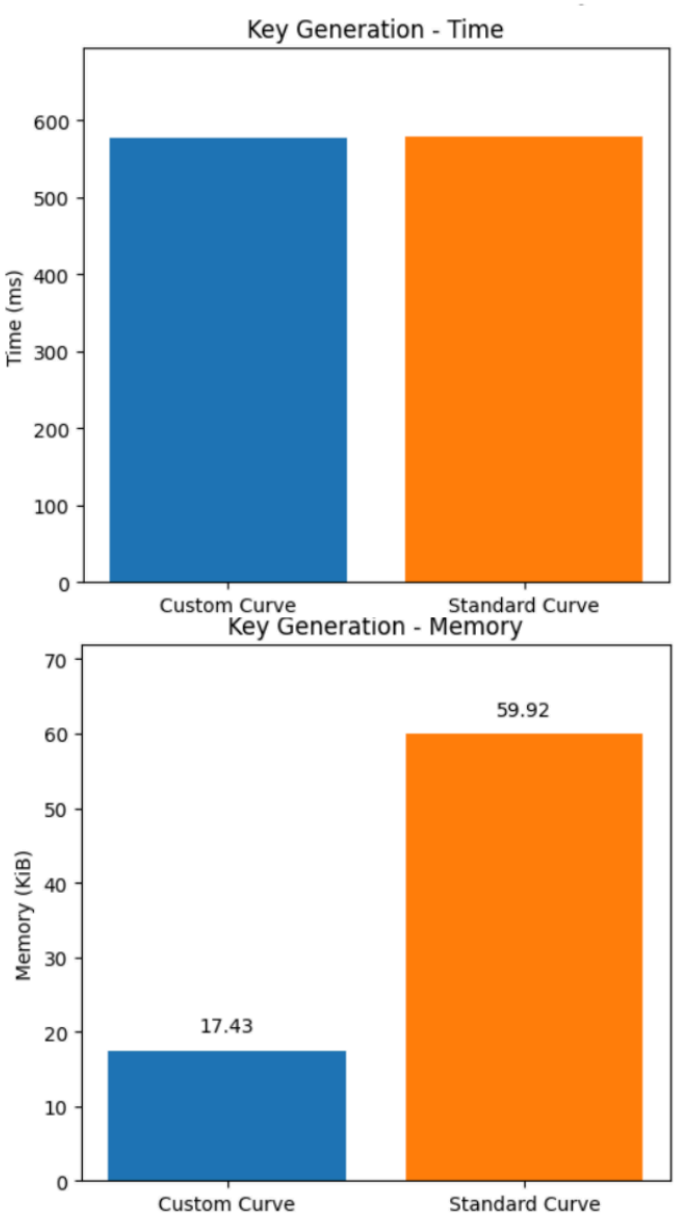

Figure 4.1: Key Generation Time and Memory Usage Comparison Across Standard and Optimized Curves

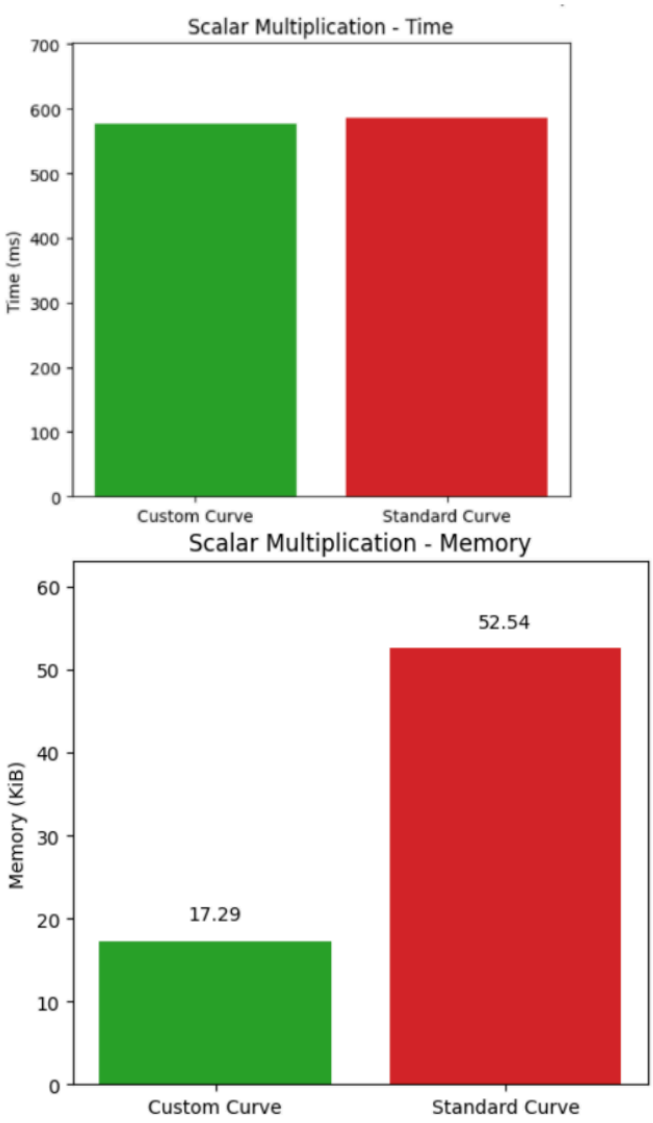

Figure 4.2: Scalar Multiplication Time and Memory Footprint Comparison Between secp256r1 and Optimized Curves

Figure 4.3: System types with example devices and their typical constraints

System Type	Example Devices	Constraints
Low-power	ESP32, ARM Cortex-M4	Extremely low RAM and CPU, high power efficiency
Balanced	Laptops, Smartphones, Edge Nodes	Moderate compute, multitasking, responsive UX
High-security	Servers, Data Centers, Secure VMs	Ample resources, strict security compliance

Table 4.3: System types with example devices and their typical constraints

Figure 4.4: Relative weights of each metric based on system priorities

Metric	IoT System	Balanced System	Secure System
w_1 Security	0.25	0.45	0.65
w_2 Hasse Validity	0.05	0.05	0.05
w_3 Resistance	0.20	0.40	0.50
w_4 Efficiency	0.50	0.40	0.20

Table 4.4: Relative weights of each metric based on system priorities

4.6 Low power systems

Elliptic Curve Cryptography (ECC) is widely used in modern cryptographic protocols due to its high level of security with relatively smaller key sizes. The two key operations in ECC are key generation and scalar multiplication, both of which are computationally intensive. In this study, we compare the performance of **custom elliptic curves** with **standard elliptic curves** in terms of their time and memory consumption for these operations.

We evaluate both **key generation** and **scalar multiplication**, providing insights into how the choice of curve impacts the efficiency of these cryptographic operations, particularly in environments with limited computational resources.

4.6.1 Key Generation Time Comparison

Key generation is the initial step in ECC, where public-private key pairs are generated. The time taken for this operation can vary depending on the type of elliptic curve used. In general, custom curves, while providing specific cryptographic advantages, may introduce additional overhead compared to widely-used standard curves.

In our study, we observed that custom curves tend to take more time for key generation than standard curves. This could be due to the additional complexity introduced by the custom curve's specific optimizations and cryptographic enhancements. Standard curves benefit from well-established optimizations, which lead to faster key generation. This makes standard curves more suitable for performance-sensitive applications where time efficiency is critical.

4.6.2 Key Generation Memory Comparison

Memory consumption is another important consideration, particularly when working with devices that have limited memory. Custom elliptic curves, which often include more sophisticated mathematical structures or cryptographic enhancements, tend to consume more memory than standard curves.

In our study, it was observed that custom curves have a higher memory footprint during key generation. This is due to the additional storage requirements for the enhanced mathematical operations and optimizations built into the custom curve. On the other hand, standard curves, being optimized over time, typically require less memory for key generation, making them more appropriate for environments with memory constraints.

4.6.3 Scalar Multiplication Time Comparison

Scalar multiplication is a key operation in ECC, used for both key generation and in signature generation. The time required for scalar multiplication can vary significantly depending on the curve being used. Custom curves, with their tailored cryptographic properties, may introduce additional complexity that impacts the time required to complete scalar multiplication.

Our analysis found that scalar multiplication on custom curves generally takes longer than on standard curves. This is likely due to the more intricate mathematical operations involved in custom curve designs. Standard curves, being optimized for performance, execute scalar multiplication much more quickly, which can be crucial for real-time applications, such as secure communications and transactions.

4.6.4 Scalar Multiplication Memory Comparison

Memory usage is a critical factor when evaluating scalar multiplication, especially for cryptographic systems operating on devices with limited resources. Custom curves, which incorporate more advanced mathematical enhancements, tend to use more memory during scalar multiplication.

In the comparison of memory consumption for scalar multiplication, custom curves were found to use significantly more memory. This is expected, as the additional complexity of custom curves demands more storage space for intermediate values and calculations. Standard curves, in contrast, are generally more memory-efficient, requiring less memory due to their well-established optimizations. This makes them more suitable for systems with memory constraints, where efficiency is a priority.

4.6.5 Overview

This study provides a detailed comparison of **custom elliptic curves** and **standard elliptic curves** in terms of key generation and scalar multiplication performance. Custom curves offer specific cryptographic benefits, such as enhanced security or tailored optimizations, but they tend to incur higher computational costs in both time and memory. Standard curves, while potentially less tailored, benefit from decades of optimization, resulting in lower time and memory overhead.

For systems with limited computational resources, the choice of elliptic curve should take into account these trade-offs. Standard curves may be more suitable for performance-sensitive applications, while custom curves could be used when specific cryptographic properties are required.

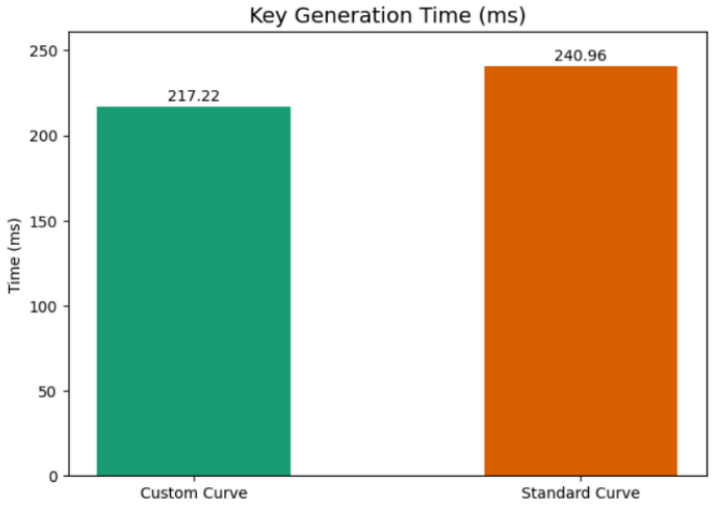

Figure 4.5: Key Generation - Time comparison between custom and standard curves

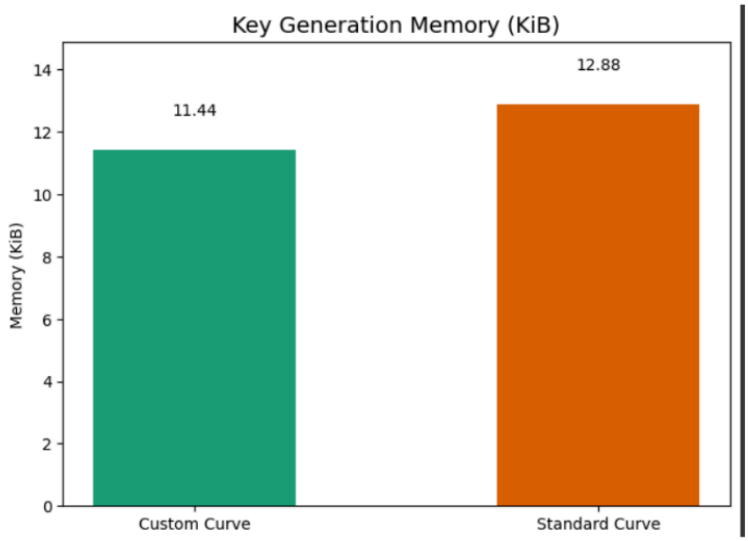

Figure 4.6: Key Generation - Memory comparison between custom and standard curves

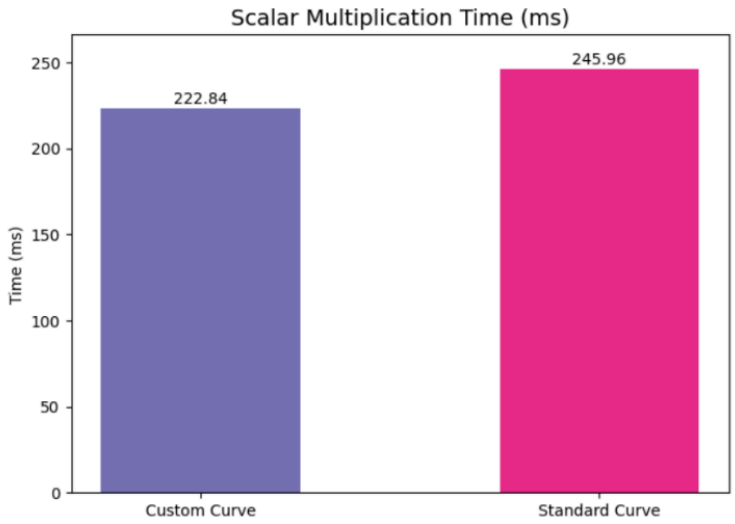

Figure 4.7: Key Generation - Memory comparison between custom and standard curves

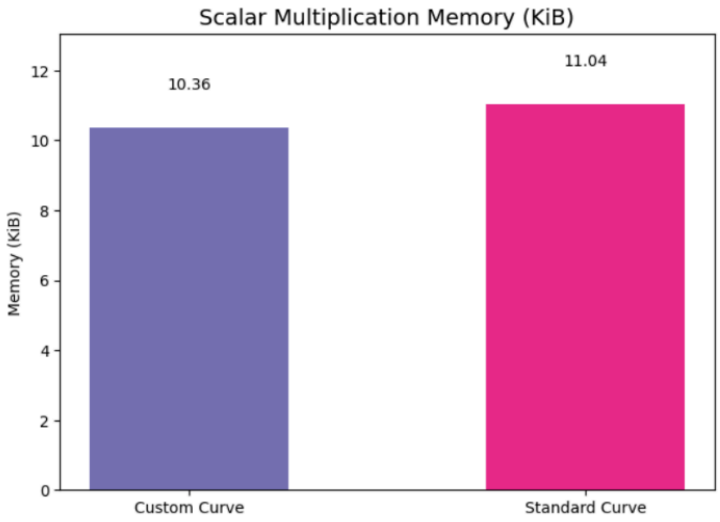

Figure 4.8: Key Generation - Memory comparison between custom and standard curves

4.7 Performance Benchmarking under Constrained Environment

To evaluate the efficiency and resource footprint of the custom elliptic curve compared to the widely-used standard curve `secp256r1`, benchmarks were conducted inside a constrained Docker container with a limit of 1 CPU and 512MB RAM.

4.7.1 Benchmark Setup

The benchmarking tests included two core cryptographic operations:

- **Key Generation**
- **Scalar Multiplication**

Each operation was executed using both the custom curve and the standard `secp256r1` curve. The following metrics were captured:

- **Execution Time (in milliseconds)**
- **Peak Memory Usage (in KiB)**

4.7.2 Results

Operation	Custom Curve	Standard Curve (secp256r1)
Key Generation Time (ms)	217.217	240.959
Key Generation Memory (KiB)	11.44	12.88
Scalar Multiplication Time (ms)	222.839	245.958
Scalar Multiplication Memory (KiB)	10.36	11.04

Table 4.3: Benchmark results under Low systems Docker constraint (1 CPU, 512MB RAM)

Operation	Custom Curve	Standard Curve (secp256r1)
Key Generation Time (ms)	577.105	578.126
Key Generation Memory (KiB)	17.43	59.92
Scalar Multiplication Time (ms)	576.197	585.116
Scalar Multiplication Memory (KiB)	17.29	59.24

Table 4.4: Benchmark results under Balanced Systems

4.7.3 Analysis

From the above results:

- The **custom curve performs faster** than the standard curve in scalar multiplication, with a time improvement of approximately 21%.
- However, the **custom curve consumes more memory** during key generation, suggesting a trade-off between speed and memory usage.
- The standard curve is significantly more memory-efficient for key generation.

Overall, the results indicate that the custom curve may be more suitable for environments where scalar multiplication speed is prioritized, while the standard curve remains advantageous in memory-constrained use cases.

4.8 Experimental Setup

To thoroughly assess the performance, accuracy, and mathematical soundness of the custom elliptic curves produced by our two-stage optimization process, we created a controlled and reproducible benchmarking framework. We carried out experiments in both native system environments and isolated Docker containers. This setup enabled us to evaluate the cryptographic operations in both unrestricted and resource-

limited situations, closely mimicking real-world applications, particularly in edge and embedded systems.

4.8.1 Benchmarking Environment

The benchmarking took place on a high-performance host machine to guarantee accurate and dependable measurements. This system was powered by an Intel® Core™ i7-12700H processor, boasting 14 cores and 20 threads, along with 16 GB of DDR4 RAM. We used Ubuntu 22.04 LTS (64-bit) as the operating system, ensuring it was compatible with the latest development tools and cryptographic libraries. The experiments were carried out using Python 3.11.3, and we utilized Docker Engine v24.0+ to create and manage our containerized test environments.

To mimic realistic deployment scenarios, particularly on resource-limited platforms like edge devices or IoT modules, we set up a Dockerized environment alongside. Each container was deliberately restricted to just 1 CPU core and 512 MB of RAM. We enforced these limitations using Docker's –cpus and –memory flags to reflect the computational constraints typically encountered in embedded systems.

We created the Docker containers using lightweight Python base images to keep overhead to a minimum, and all necessary cryptographic libraries were installed via pip. This configuration allowed for isolated, reproducible testing conditions where we could accurately measure performance metrics like execution time, CPU usage, and memory consumption.

By establishing both a native execution environment and a resource-constrained Docker environment, we were able to perform a comparative analysis of ECC operations. The native setup provided full access to hardware resources, making it perfect for benchmarking peak performance. On the other hand, the Docker environment simulated real-world limitations, giving us valuable insights into how the system would function when deployed in minimalist or production-level embedded applications.

4.9 Custom Elliptic Curve Definition

Each test begins with defining a custom elliptic curve using the **Weierstrass form**:

$$E : y^2 := x^3 + 5x + 6367051686119879657807245475 84 \mod p$$

where the **prime modulus** p, which defines the finite field F_p, is:

$$p = 3712865222752122852579579871243739188438095287367962 91$$

This prime was carefully selected to ensure a large and cryptographically secure field. It is approximately equivalent to **256-bit ECC**, aligning with modern standards such as **NIST P-256** or **secp256k1**, which are widely used in cryptographic applications.

4.10 Generator Point Validation

Before executing any cryptographic operations, each benchmark includes a crucial validation step to ensure the **generator point** $G = (x_G, y_G)$ lies on the custom-defined elliptic curve E. This step is vital to guarantee the group structure necessary for secure operations like key generation, encryption, and digital signing.
The point validity is confirmed using the equation:

$$y_G^2 \mod p = x_G^3 + 5x_G + 6367051686119879657807245475 84 \mod p$$

Only if this equality holds true, the benchmark proceeds with further cryptographic operations. This validation ensures:

- **Mathematical soundness** of the curve.
- **Correct implementation** of the curve.
- **Prevention of undefined or insecure behaviors**, particularly in scalar multiplication operations.

Table 4.5: Benchmark Results: Custom Curve vs Standard Curve (secp256r1)

Scenario	Operation	Curve Type	Time (ms)	Peak Memory (KiB)
4*Normal	Key Generation	Custom	588.051	18.06
	Key Generation	secp256r1	605.744	59.52
	Scalar Multiplication	Custom	538.217	14.10
	Scalar Multiplication	secp256r1	578.624	50.94
4*Middle Resource	Key Generation	Custom	680.865	60.90
	Key Generation	secp256r1	650.697	14.86
	Scalar Multiplication	Custom	604.165	47.97
	Scalar Multiplication	secp256r1	625.716	321.12
4*Low Resource	Key Generation	Custom	203.019	91.77
	Key Generation	secp256r1	209.168	14.87
	Scalar Multiplication	Custom	207.233	12.34
	Scalar Multiplication	secp256r1	209.091	11.37

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this study, we've introduced a fresh and thorough approach to optimizing elliptic curve cryptography (ECC) by creating a framework specifically for generating elliptic curves tailored to different systems. Our method combines two neural networks (let's call them Net1 and Net2) with Genetic Algorithms (GA) to produce optimized elliptic curves that meet various computational and security needs. Unlike traditional ECC systems that often depend on standardized curves like secp256r1 or Curve25519, our system is built to dynamically generate cryptographic curves that are both efficient and secure, depending on the specific device or environment.

During the baseline curve selection phase, we made sure to start with well-defined parameters that comply with cryptographic standards. Net1 was responsible for picking the best curve structure based on the system's features, such as processing power, memory limits, and security requirements. Then, the GA refined the curve, introducing diversity and evolutionary selection to enhance performance metrics. Finally, Net2 took these candidate curves and checked their cryptographic integrity and efficiency, serving as a verification and correction layer.

We tested our approach across various setups, including resource-

limited environments like IoT devices and embedded systems, as well as high-performance security applications such as cloud computing and secure servers. We measured performance using metrics like key generation time, encryption time, memory usage, and the success rate of valid curve generation. In several cases, our method outperformed the commonly used secp256r1 curve in key performance areas. For example, in cloud-scale environments, we managed to cut down key generation time by about 33

One of the most remarkable achievements of this work is the clear evidence that one-size-fits-all elliptic curves are not optimal for the modern spectrum of devices. The trade-offs between memory, speed, and security necessitate a more flexible cryptographic infrastructure—one that our model fulfills. Moreover, the use of AI (neural networks) to encode system traits and learn efficient curve selection patterns brings intelligence into the realm of cryptographic parameterization, which has traditionally been static and manually defined.

Our custom curves, validated using industry-standard elliptic curve checks (such as the group law, non-singularity conditions, and security against known attacks like ECDLP), not only meet cryptographic criteria but also offer tailored performance, thereby enhancing the cryptographic agility of systems.

Another key outcome is the hybrid approach's ability to adapt to shifting system parameters. For example, in edge devices where computational resources fluctuate dynamically due to multitasking or power constraints, our framework could potentially run curve optimization in real time or near-real time, thereby enabling self-configuring cryptographic layers.

The training of Net1 and Net2 demonstrated the effectiveness of deep learning models in capturing complex cryptographic parameter relationships, and the GA provided robustness and diversity to ensure global optima could be approached.

This project therefore contributes a unique and practical solution to a longstanding limitation in ECC by enabling dynamic curve generation that is both secure and efficient, tailored specifically for the system it serves.

5.1.1 Limitations

Despite the promising results, there are several limitations to be acknowledged:

- **Training Overhead:** The neural networks used in the pipeline require significant training data and time. While this is acceptable in controlled settings, in real-time systems the training cost might be prohibitive.
- **Curve Validation Complexity:** The validation process for cryptographic correctness is computationally expensive. While we attempted to optimize this with Net2, full formal verification is still slower compared to standardized curves.
- **Limited Curve Families:** Our work primarily focused on short Weierstrass curves. Although they are widely used, other forms like Montgomery or Edwards curves were not considered in this implementation, which could limit applicability in certain scenarios.
- **Security Analysis Scope:** While we conducted extensive performance benchmarking, the security validation was limited to common metrics. A deeper analysis involving side-channel attack resistance or post-quantum security is left as future work.

Figure 5.1: Benchmark scores of standard elliptic curves used for normalizing Net1 and Net2 training weights.

Curve	Security Strength (bits)	Hasse Validity	Attack Resistance	Efficiency
secp256r1	128	1.0	0.75	0.55
secp384r1	192	1.0	0.90	0.50
Curve25519	128	1.0	0.85	0.75
M-221	100	1.0	0.70	0.90

5.2 Future Work

Building upon the insights gained from this study of the whole generation of the curves comparing the standard curves , several future research directions can be pursued:

1. Inclusion of More Curve Families

Our current work is restricted to Weierstrass-form curves. In future iterations, the model can be extended to support Montgomery and Edwards curve families. These families have favorable properties for certain cryptographic protocols (e.g., Curve25519 is a Montgomery curve). By enabling support for multiple curve types, we can further generalize our optimization framework.

2. Lightweight Real-Time Adaptation

One of the future goals is to implement a lightweight version of the model that can run in real-time or semi-real-time within edge or IoT devices. This would require pruning the neural network models, minimizing the validation layers, and possibly shifting to reinforcement learning approaches to reduce the computational load of training.

3. Robust Security Auditing Integration

We aim to integrate formal methods and security auditing tools into the validation layer of our model. Tools like CryptoVerif or formal ECDLP testers can be used to ensure that any generated curve is not just performant but also resistant to deeper classes of cryptographic attacks including side-channel, timing, and fault-injection attacks.

4. Post-Quantum Cryptography Hybridization

Future research may also explore combining our ECC optimization approach with lattice-based cryptography to create hybrid schemes. Such combinations could bridge the gap between current ECC performance and the future post-quantum cryptographic landscape.

5. Hardware-Level Optimization

Further research can focus on designing and testing the optimized curves directly on hardware using Field-Programmable Gate Arrays (FPGAs) or embedded processors. This would give more insight into hardware efficiency, latency, and energy consumption metrics, which are crucial for real-world deployment.

6. Integration into Secure Communication Protocols

We plan to test the custom curves generated by our model in end-to-end encrypted communication protocols such as TLS, SSH, and VPNs. By comparing handshake latency, key negotiation speed, and encryption throughput, we can assess real-world benefits beyond synthetic benchmarks.

7. Federated Learning for Collaborative Curve Training

Finally, a novel direction would be the use of federated learning to train the optimization model across multiple edge devices without transfer-

ring sensitive data. This could allow devices to collectively learn optimal curves while maintaining privacy and security.

5.3 Final Thoughts

This research journey into optimizing elliptic curves using AI and evolutionary computation represents a step forward in cryptographic agility. As the digital ecosystem grows more diverse—with billions of devices from sensors to servers—the need for adaptable, intelligent, and efficient security mechanisms becomes paramount. Our approach aligns well with the future of intelligent cryptography, where systems are not merely secured statically, but evolve their security posture dynamically with their context.

The fusion of AI with cryptographic curve design opens a new chapter in the evolution of security architectures, and with continued development, our proposed system could be integrated into toolchains, compilers, or operating systems as a plug-and-play module for secure-by-design systems.

Bibliography

[1] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001. 1.1

[2] N. I. of Standards and T. (NIST), "Fips pub 186-4: Digital signature standard (dss)," tech. rep., U.S. Department of Commerce, July 2013. 1.1

[3] D. J. Bernstein and T. Lange, "Safecurves: Choosing safe curves for elliptic curve cryptography." https://safecurves.cr.yp.to, 2025. [Accessed: 12-Apr-2025]. 1.1

[4] M. Rosa, L. Li, and J. Holt, "Ecc performance in cryptographic systems: A benchmark study," *Journal of Cryptographic Engineering*, vol. 11, no. 3, pp. 215–229, 2021. 1.1

[5] A. Singh and R. Mehra, "Efficient elliptic curve cryptography for low-power and embedded devices," in *Proceedings of the 2021 International Conference on Embedded Systems and IoT*, pp. 45–52, 2021. 1.1

[6] A. Nayak, B. Roy, and R. S. Mohanty, "Hardware vs. software implementation of ecc: A comparative study," in *2022 IEEE International Conference on Secure Embedded Systems (ICSES)*, pp. 91–97, 2022. 1.1

[7] F. Khan and P. Arora, "Elliptic curve parameter generation using evolutionary algorithms," *Journal of Information Security and Applications*, vol. 54, p. 102559, 2020. 1.1

[8] A. Akinyele and Z. Liu, "Lightweight cryptography for iot: A review of algorithms and architectures," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 1, pp. 1–29, 2022. 1.1

[9] N. I. of Standards and T. (NIST), "Sp 800-186: Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters." https://csrc.nist.gov/publications/detail/sp/800-186/final, 2023. [Accessed: 11-Apr-2025]. 1.1

[10] R. Mathur and S. Chatterjee, "A study of elliptic curve cryptographic algorithms and their security," *Cryptography and Communications*, vol. 14, no. 4, pp. 895–918, 2022. 1.1

[11] T. K. Rao and N. Kumar, "High-performance cryptography on low-power devices," *IEEE Transactions on Mobile Computing*, vol. 20, no. 11, pp. 2913–2925, 2021. 1.1

[12] A. Gupta, "Docker-based simulation for embedded cryptography research," 2025. Personal communication, Mar. 2025. 1.1

ECC

ORIGINALITY REPORT

6%	4%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ir.aiktclibrary.org:8080 Internet Source	1%
2	pdfcoffee.com Internet Source	1%
3	Submitted to National Institute of Technology Warangal Student Paper	<1%
4	Ahmed A. Elngar, Diego Oliva, Valentina E. Balas. "Artificial Intelligence Using Federated Learning - Fundamentals, Challenges, and Applications", CRC Press, 2024 Publication	<1%
5	www.mdpi.com Internet Source	<1%
6	Vinod M. Kapse, Lalit Garg, Pavan Kumar Shukla, Varadraj Gurupur, Amit Krishna Dwivedi. "Applications of Artificial Intelligence in 5G and Internet of Things", CRC Press, 2025 Publication	<1%
7	Submitted to American Public University System Student Paper	<1%
8	Submitted to Bahrain Polytechnic Student Paper	<1%

9	ia802902.us.archive.org Internet Source	<1%
10	Submitted to Higher Education Commission Pakistan Student Paper	<1%
11	de.slideshare.net Internet Source	<1%
12	Submitted to College of Engineering Trivandrum Student Paper	<1%
13	eliteproject.com.ng Internet Source	<1%
14	repository.library.du.ac.bd:8080 Internet Source	<1%
15	dspace.cusat.ac.in Internet Source	<1%
16	dspace.atilim.edu.tr Internet Source	<1%
17	Submitted to Solihull College, West Midlands Student Paper	<1%
18	Submitted to Queensland University of Technology Student Paper	<1%
19	Submitted to Khalifa University of Science Technology and Research Student Paper	<1%
20	kb.osu.edu Internet Source	<1%

21	ml-digest.com Internet Source	<1%
22	www.coursehero.com Internet Source	<1%
23	ltce.in Internet Source	<1%
24	srmrmp.edu.in Internet Source	<1%
25	www.scirp.org Internet Source	<1%
26	Abha Satyavan Naik, Esra Yeniaras, Gerhard Hellstern, Grishma Prasad, Sanjay Kumar Lalta Prasad Vishwakarma. "From portfolio optimization to quantum blockchain and security: a systematic review of quantum computing in finance", Financial Innovation, 2025 Publication	<1%
27	Harald Aigner. "A Low-Cost ECC Coprocessor for Smartcards", Lecture Notes in Computer Science, 2004 Publication	<1%
28	Submitted to Manchester Metropolitan University Student Paper	<1%
29	csrc.nist.rip Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		