

M-ary Precomputation-Based Accelerated Scalar Multiplication Algorithms for Enhanced Elliptic Curve Cryptography

Tongxi Wu^{a,*}, Xufeng Liu^{a,*}, Jin Yang^{a,**}, Yijie Zhu^b, Shunyang Zeng^a,
Mingming Zhan^a

^aCollege of Cyber Science and Engineering, Sichuan University, Chengdu 610207, China

^bInstitute of Software, Chinese Academy of Sciences and University of Chinese Academy of Sciences, Beijing 100190, China

Abstract

Efficient scalar multiplication is critical for enhancing the performance of elliptic curve cryptography (ECC), especially in applications requiring large-scale or real-time cryptographic operations. This paper proposes an M-ary precomputation-based scalar multiplication algorithm, aiming to optimize both computational efficiency and memory usage. The method reduces the time complexity from $\Theta(Q \log p)$ to $\Theta\left(\frac{Q \log p}{\log Q}\right)$ and achieves a memory complexity of $\Theta\left(\frac{Q \log p}{\log^2 Q}\right)$. Experiments on ElGamal encryption and NS3-based communication simulations validate its effectiveness. On secp256k1, the proposed method achieves up to a 59% reduction in encryption time and 30% memory savings. In network simulations, the binary-optimized variant reduces communication time by 22.1% on secp384r1 and simulation time by 25.4% on secp521r1. The results demonstrate the scalability, efficiency, and practical applicability of the proposed algorithm. The source code will be publicly released upon acceptance.

Keywords: Scalar Multiplication, ECC, Cryptology, Precomputation

*These authors contributed equally to this work.

**Corresponding author. Email: wtx2021141230137@stu.scu.edu.cn

1. Introduction

Information technology has become one of the fundamental pillars of modern society, enabling innovations such as distributed computing, databases, and blockchain systems [1, 2]. As the value of information systems increases, they have become prominent targets for cyberattacks. Cryptographic techniques are essential to protect these systems, ensuring confidentiality, integrity, and availability. In particular, asymmetric cryptographic algorithms are widely adopted due to their advantages in key management and security strength [3]. Furthermore, they provide critical digital signature capabilities [4], which are foundational to applications such as cryptocurrencies and decentralized finance.

Elliptic curve cryptography (ECC) [5] is widely employed in information system encryption because of its robust security, high efficiency, and compact key size [6]. ECC has garnered attention for its effectiveness in various applications, including financial transactions, network security, and data protection[7]. In these applications, enhancing the efficiency of ECC is crucial, particularly in scenarios that require quick responses and high security. A key aspect of enhancing ECC efficiency is scalar multiplication [8], a fundamental operation in ECC-based protocols [9]. Efficient scalar multiplication reduces latency and power consumption, thereby facilitating faster and more secure data transmission.

One of the main challenges in Elliptic Curve Cryptography (ECC) lies in the high computational cost of scalar multiplication. As the scale and computational demands of modern information systems continue to grow [7], the need for more efficient and secure scalar multiplication algorithms becomes increasingly critical [10]. Existing approaches often struggle to balance computational speed and memory usage, particularly in resource-constrained environments.

To address these challenges, this paper proposes an accelerated scalar multiplication algorithm based on M-ary precomputation. The proposed method reduces the time complexity from $\Theta(Q \log p)$ to $\Theta\left(\frac{Q \log p}{\log Q}\right)$ while achieving flexible memory efficiency through structured precomputation. This work aims to enhance the performance and scalability of ECC operations across a broad range of cryptographic applications.

The key contributions of this research are as follows:

1. **Innovative Scalar Multiplication Algorithm:** We propose a flexi-

ble M-ary precomputation-based scalar multiplication algorithm for Elliptic Curve Cryptography (ECC), achieving significant improvements in both time and memory efficiency. Compared to conventional methods such as the sliding window and fixed-base comb algorithms, the proposed method achieves linear memory scaling and better adapts to arbitrary scalar sizes.

2. Theoretical Improvements in Time and Memory Complexity:

Our method reduces the time complexity from $\Theta(Q \log p)$ to $\Theta\left(\frac{Q \log p}{\log Q}\right)$

and the memory complexity to $\Theta\left(\frac{Q \log p}{\log^2 Q}\right)$ through structured precomputation and sparse storage. This provides an asymptotic advantage over traditional precomputation techniques.

- 3. Significant Performance Gains:** Experimental evaluations based on ElGamal encryption and NS3 communication simulations demonstrate the practical effectiveness of the proposed method. Specifically, the algorithm achieves up to a 59% reduction in encryption time and a 30% reduction in peak memory usage compared to baseline methods when $Q = 1000$. Moreover, the binary-optimized variant achieves a 22.1% reduction in total communication time on secp384r1 and a 25.4% reduction in overall simulation time on secp521r1, highlighting its scalability across different elliptic curves and workloads.

2. Related work and preliminaries

2.1. Related Work

ECC has been extensively studied for its efficiency and security advantages, particularly in resource-constrained settings. Existing research can be broadly categorized into three areas: optimizations of ECC algorithms, applications of ECC in resource-constrained settings, and lightweight cryptographic techniques.

Scalar multiplication is the most computationally intensive operation in elliptic curve cryptography (ECC) and has been the subject of extensive optimization research [8, 9]. Traditional methods such as Double-and-Add [11] and NAF-based algorithms [12] focus on reducing the number of point additions. More advanced techniques, including 2^k -ary methods [13], Sliding Window algorithms [14, 15], Montgomery Ladder [15, 16], Fixed-Base Comb [17], and Window τ -NAF representations [18], further enhance computational efficiency or side-channel resistance. While these approaches significantly im-

prove scalar multiplication performance, they often encounter scalability and memory trade-offs when handling large numbers of scalar operations. This paper addresses these limitations by proposing an M-ary precomputation-based algorithm that simultaneously improves time complexity and memory efficiency, providing a flexible and scalable solution for ECC applications.

Research Area	Related Work	Key Contributions
ECC Algorithm Optimization	Scalar Multiplication	Reduce the number of point additions and multiplications [11, 12]
	Precomputation Techniques	Improve performance but require significant memory [14, 15]
	Homomorphic Encryption	Focus on high-performance computing [19]
	GPU-Accelerated Computations	Target high-performance computing rather than IoT [20]
ECC Applications in Resource-constrained Settings	Secure Communication	ECC's Error Correction and Side-Channel Attack Resistance [21]
	Decentralized Federated Learning	Enhances security and scalability in IoT [22]
	Physical Layer Security	Improves data security in IoT [23]
	Machine Learning in ECC	Anomaly detection and authentication enhance ECC's robustness [24, 25]
Lightweight Cryptography	Lightweight Protocols	Optimize resource utilization while maintaining strong encryption [7]
	Tensor-Based Models	Enhance reliability and robustness in hybrid IoT systems [26]
	Standard ECC Limitations	Standard ECC implementations struggle with IoT constraints [5, 27]
	Lightweight ECC Protocols	Improve scalability and efficiency for IoT applications [28]

Table 1: Summary of Related Work in ECC and IoT Applications

Recent advancements in ECC optimizations, including homomorphic encryption [19] and GPU-accelerated computations [20], primarily focus on high-performance computing. However, these methods demand additional resources, making them less suitable for resource-constrained settings. Hardware-level optimizations [29] lack portability, whereas algorithm-level optimizations are more beneficial for practical applications.

ECC is widely employed in communications in resource-constrained settings because of its small key size and high computational efficiency [30]. High-impact applications include decentralized federated learning (DFL) [22], physical layer security [23], and secure multimedia transmission techniques, such as 3D scrambling [31]. Machine learning methods, such as anomaly detection and authentication [24, 25], have further enhanced the robustness of ECC-based systems in resource-constrained settings.

Despite these advances, the computational limitations of resource-constrained settings frequently result in performance bottlenecks [10, 32, 33]. Traditional ECC implementations face challenges in balancing security and real-time communication requirements in high-frequency IoT data exchanges [34, 27].

Lightweight cryptography has emerged as a promising solution to the efficiency challenges in resource-constrained settings [7]. These protocols optimize resource utilization while maintaining strong encryption, thereby ensuring scalability and robustness [28]. Techniques such as tensor-based models have been proposed to enhance network reliability and robustness in hybrid resource-constrained systems [26].

However, existing lightweight cryptographic protocols typically rely on standard ECC implementations, which are not optimized for the specific constraints of resource-constrained devices [5, 27]. The need for fundamental improvements in ECC algorithms remains largely unaddressed, resulting in a significant gap in the literature.

Although ECC has undergone multiple optimizations, the computational cost remains unacceptable when a large number of scalar multiplications are required [35]. No existing method can identify the path with the shortest step length for a specific computational task while balancing computational and space costs. Hardware-optimized scalar multiplication demonstrates scalability and robustness in resource-constrained settings; however, its reliance on traditional methods limits efficiency [10, 32, 36]. Memory-intensive approaches, such as Sliding Window techniques [14, 15], are unsuitable for IoT devices with limited storage capacity. This study addresses these gaps by introducing a novel M-ary Precomputation-Based Accelerated Scalar Multiplication algorithm. By reducing both time complexity and memory usage, the proposed method offers a scalable and efficient solution for secure information communication.

2.2. Preliminaries

In this section, we present the core principles and operations related to elliptic curves, as well as the scalar multiplication algorithm and its optimization techniques.

2.2.1. Elliptic Curves [37]

An elliptic curve over a finite field \mathbb{F}_p is of this form:

$$E_p(a, b) = \left\{ (x, y) \mid x, y \in \mathbb{F}_p, y^2 \equiv x^3 + ax + b \pmod{p} \right\} \cup \{O\}. \quad (1)$$

Where p is a prime number, and $a, b \in \mathbb{F}_p$ satisfy the condition $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, the point O is referred to as the point at infinity. When there is no need to specify a and b , the elliptic curve $E_p(a, b)$ can also be denoted as $E(\mathbb{F}_p)$. The field \mathbb{F}_p is called the base field of the elliptic curve $E(\mathbb{F}_p)$.

The process of adding points on an elliptic curve is defined as follows.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points in $E_p(a, b) \setminus \{O\}$. Then, the addition of P and Q is defined as follows:

$$P + Q = \begin{cases} O, & x_1 = x_2, y_1 = -y_2, \\ (x_3, y_3), & \text{otherwise.} \end{cases} \quad (2)$$

where

$$x_3 = \lambda^2 - x_1 - x_2, \quad (3)$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \quad (4)$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q, \\ \frac{3x_1^2 + a}{2y_1}, & P = Q. \end{cases} \quad (5)$$

2.2.2. Scalar Multiplication Algorithms [16]

Let $n \in \mathbb{N}$ and $P \in E_p(a, b)$. Then, the scalar multiplication of the non-negative integer n with the point P is defined as follows:

$$nP = \begin{cases} O, & n = 0, \\ (n-1)P + P, & n \geq 1. \end{cases} \quad (6)$$

Table 2: Comparative Analysis of Time and Space Complexities for Related Elliptic Curve Scalar Multiplication Algorithms.

Algorithm	Time Complexity	Space Complexity
Double-and-add [11]	$\Theta(Q \log p)$	$\Theta(1)$
NAF-based [12]	$\Theta(Q \log p)$	$\Theta(1)$
2^k -ary [13]	$\Theta(Q \log p)$	$\Theta(1)$
Sliding Window [14, 15]	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$
Montgomery Ladder [14, 15]	$\Theta(Q \log p)$	$\Theta(1)$
Fixed-Base Comb [17]	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$
Window τ -NAF [18]	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$

Next, we introduce seven widely adopted optimization algorithms for elliptic curve scalar multiplication: the double-and-add algorithm [11], the NAF-based scalar multiplication algorithm [12], the 2^k -ary method [13], the Sliding Window algorithm [14, 15], the Montgomery Ladder technique [15, 16], the Fixed-Base Comb algorithm [17], and the Window τ -NAF precomputation scheme [18]. These algorithms have been extensively employed to enhance the computational efficiency of scalar multiplication in elliptic curve cryptography (ECC).

Among them, the Sliding Window algorithm [14, 15] significantly reduces the number of point additions by leveraging a precomputed table of odd multiples of the base point. The Montgomery Ladder algorithm [15, 16] offers a constant-time implementation that is inherently resistant to side-channel attacks, thus achieving a favorable trade-off between security and efficiency. The Fixed-Base Comb algorithm [17] partitions the scalar into multiple columns based on a fixed base, enabling parallel computation and fast table lookups when the base point is constant. The Window τ -NAF precomputation scheme [18] combines the efficiency of non-adjacent form (NAF) representations with τ -adic expansions in Koblitz curves, further reducing the number of required operations through strategic windowing and precomputation.

These algorithms are widely utilized to improve the efficiency and security of scalar multiplication in ECC. Table 2 summarizes the time and space complexities of these algorithms.

2.2.3. Methods

Double-and-add Algorithm [11]. The Double-and-Add Algorithm is a fundamental method for scalar multiplication in elliptic curve cryptography (ECC). Its simplicity has significantly contributed to the development of scalar multiplication algorithms, with many subsequent algorithms being optimizations of this approach.

The core of Double-and-add Algorithm in elliptic curve cryptography is the following identity:

$$k = 2 \left\lfloor \frac{k}{2} \right\rfloor + k \bmod 2. \quad (7)$$

This transformation reduces the computation of kP to that of $\lfloor \frac{k}{2} \rfloor P$. Consequently, the time complexity $T(k)$ for computing the scalar multiplication kP follows the recursive formula:

$$T(k) = T\left(\frac{k}{2}\right) + \Theta(1). \quad (8)$$

NAF Based Scalar Multiplication Algorithm [12]. The Double-and-Add Algorithm [11] is a basic yet effective method for scalar multiplication; however, its performance can often be improved. This has led to the development of scalar multiplication algorithms based on the Non-Adjacent Form (NAF) [12], which optimize the computation through a novel representation of integers.

The NAF representation expresses the scalar k in a way that minimizes the number of required point additions. Specifically, to compute the scalar multiplication kP , we represent k in its NAF as:

$$k = \sum_{i=0}^{d-1} a_i 2^i. \quad (9)$$

The scalar multiplication is then computed as:

$$kP = \sum_{i=0}^{d-1} a_i (2^i P). \quad (10)$$

Denoting $A_i = 2^i P$ for $i \geq 0$, the scalar multiplication can be rewritten as:

$$kP = \sum_{i=0}^{d-1} a_i A_i. \quad (11)$$

The NAF representation typically results in fewer elliptic curve point additions compared to the Double-and-Add method, leading to improved practical efficiency in scalar multiplication.

2^k -ary [13]. The 2^k -ary algorithm [13] is used to compute scalar multiplication mP . The algorithm begins by selecting a positive integer k and representing the scalar coefficient m as a base- 2^k number:

Where $0 \leq a_i < 2^k$ and $a_{d-1} > 0$, the algorithm defines the intermediate values A_j as follows:

$$A_j := \sum_{i=d-j}^{d-1} a_i 2^{k(i-(d-j))} P, 0 \leq j \leq d. \quad (12)$$

With $A_0 = O$ and $A_d = mP$. For $1 \leq j \leq d$, we have:

$$A_j = 2^k A_{j-1} + a_{d-j} P. \quad (13)$$

To simplify the computation, each coefficient a_i can be expressed as:

$$a_i = u_i \cdot 2^{s_i}, \quad (14)$$

where u_i is an odd integer (i.e., u_i is not divisible by 2), and s_i is a non-negative integer that represents the power of 2 in the factorization of a_i .

The steps involve doubling A_{j-1} a total of $k - s_{d-j}$ times to compute $2^{k-s_{d-j}} A_{j-1}$, retrieving $u_{d-j} P$ from a precomputed table, and adding this to the doubled result. The final result is then doubled s_{d-j} times to obtain A_j .

The advantage of this method lies in reducing the number of point additions (excluding doublings) required on the elliptic curve, thereby enhancing efficiency. In practice, setting $k = 5$ is often optimal for performance.

Sliding Window algorithm [14, 15]. The Sliding Window algorithm [14, 15] is an optimization technique in elliptic curve cryptography (ECC) designed to enhance the efficiency of scalar multiplication, computing $Q = kP$, where k is the scalar and P is a point on the elliptic curve. This algorithm partitions k into fixed-size windows of ω bits, with each window w_i representing a segment of the scalar.

The algorithm processes k in ω -bit windows by precomputing and storing each odd multiple $(2j + 1)P$ in a table, which reduces redundant computations. During execution, for a given window w_i , the algorithm combines

point doubling and table lookups, performing the operation:

$$Q \leftarrow 2^\omega Q + P_{w_i}, \quad (15)$$

where P_{w_i} is the precomputed value for the window w_i , and $2^\omega Q$ represents ω consecutive doublings. This approach significantly reduces the number of point additions, balancing memory usage with computational complexity.

The Sliding Window algorithm [14, 15] requires $O(2^{\omega-1})$ storage for pre-computed values and reduces the average number of point additions from $\lfloor \log_2 k \rfloor$ (as seen in the Double-and-Add algorithm) to approximately $\lfloor \log_2 k \rfloor / \omega$. By tuning ω , the algorithm can adapt to various performance and hardware constraints, making it particularly effective for scalars with large bit lengths.

Montgomery Ladder algorithm [16]. The Montgomery Ladder algorithm [16] is an efficient and secure method for scalar multiplication in elliptic curve cryptography (ECC), particularly noted for its resistance to side-channel attacks. This algorithm computes the scalar multiplication $Q = kP$ by maintaining a consistent operation flow, regardless of the scalar k , which mitigates simple power analysis (SPA) vulnerabilities.

In the Montgomery Ladder, the scalar k is processed bit-by-bit from the most significant to the least significant bit. At each iteration, two points, R_0 and R_1 , are updated to satisfy the invariant $R_1 - R_0 = P$. This ensures that only differential point additions and doublings are used, enhancing security.

The algorithm is compatible with various elliptic curve coordinate systems, including Jacobian and projective coordinates, optimizing performance based on specific hardware or software constraints. Its robustness against side-channel attacks, along with its adaptability to different elliptic curve forms, underscores its importance in modern ECC implementations.

The operational flow of Sliding Window algorithm and Montgomery Ladder algorithm is illustrated in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Sliding Window Algorithm

Input: $P \in E(\mathbb{F}_q)$, $k = \sum_{i=0}^{l-1} k_i 2^i$, window size ω
Output: $Q = kP$
Precompute P_{2i+1} , $Q \leftarrow O$, $i \leftarrow l - 1$
while $i \geq 0$ **do**
 if $k_i = 0$ **then**
 $Q \leftarrow 2Q$, $i \leftarrow i - 1$
 else
 Find max t s.t. $i - t + 1 \leq \omega$, $k_t = 1$
 $h_i \leftarrow (k_i \dots k_t)_2$
 $Q \leftarrow [2^{i-t+1}]Q + P_{h_i}$
 $i \leftarrow t - 1$
 end if
end while
Return: Q

Algorithm 2 Montgomery Ladder Algorithm

Input: $P \in E(\mathbb{F}_q)$, scalar $k = \sum_{i=0}^{n-1} k_i 2^i$
Output: $Q = kP$
 $R_0 \leftarrow O$, $R_1 \leftarrow P$
for $i = n - 1$ **downto** 0 **do**
 if $k_i = 0$ **then**
 $R_1 \leftarrow R_0 + R_1$
 $R_0 \leftarrow 2R_0$
 else
 $R_0 \leftarrow R_0 + R_1$
 $R_1 \leftarrow 2R_1$
 end if
end for
Return: R_0

Fixed-Base Comb algorithm [17]. The Fixed-Base Comb algorithm [17] represents the scalar k in width- ω Non-Adjacent Form (NAF) and divides it into $\omega \times v$ blocks, processed from top to bottom and right to left.

First, the scalar k is divided into $a = \lceil l/\omega \rceil$ blocks of size ω , with padding added if necessary. Each block K_d consists of ω bits, allowing k to be expressed as a series of these blocks. The scalar multiplication kP can thus be computed as follows:

$$kP = K_{a-1}K_{a-2} \dots K_0P = \sum_{j=0}^{v-1} \sum_{t=0}^{b-1} (K_{jb+t} 2^{t\omega})P, \quad (16)$$

where K_{jb+t} represents the block in width- ω NAF representation.

To optimize performance, values $G[j][sd]$ are precomputed for necessary indices, allowing kP to be expressed in a more efficient form:

$$kP = \sum_{t=0}^{b-1} 2^{t\omega} \left(\sum_{j=0}^{v-1} G[j][I_j, t] \right). \quad (17)$$

This algorithm effectively balances the average and worst-case densities of non-zero digits in the width- ω NAF representation, contributing to its overall

efficiency. By optimizing the representation and processing of the scalar k , this approach significantly enhances the performance of scalar multiplication in elliptic curve cryptography.

Pre-computation Scheme of Window τ NAF [18]. Yu and Xu revisit the pre-computation scheme for the window τ NAF (non-adjacent form) method, specifically designed for Koblitz curves. Their innovative approach enhances the efficiency of scalar multiplication by introducing new algebraic operations known as $\mu\bar{\tau}$ -operations [18], which utilize the complex conjugate of the Frobenius map. This improvement surpasses traditional methods, resulting in fewer field operations during the pre-computation phase and facilitating faster scalar multiplication.

This algorithm uses a unified pre-computation scheme that maintains efficiency across different configurations of Koblitz curves, particularly for curves E_0 and E_1 . Their results reveal time complexities of $6M + 6S$, $18M + 17S$, and $44M + 32S$ for window widths of 4, 5, and 6, respectively, when $a = 0$. These costs are approximately twice as efficient compared to existing pre-computation techniques.

Overall, this innovative pre-computation method not only enhances computational efficiency but also suggests the potential for further advancements in elliptic curve cryptography, particularly for other types of curves defined over F_3^m or F_q^m for primes $q \geq 5$. Thus, their work significantly contributes to both theoretical understanding and practical implementations in the field.

3. Our Proposed Algorithm

In this section, we present an optimized approach for scalar multiplication based on M-ary precomputation. This method is designed to accelerate the computation of multiple scalar multiplications on elliptic curves, significantly improving performance over previous algorithms. We will outline the key principles behind the algorithm, its time complexity, and the optimizations that minimize computational overhead. Recent advances in the theoretical understanding of time-space tradeoffs for function inversion [38] suggest that careful balancing of precomputation storage and online computation cost is crucial for achieving optimal performance. Motivated by these insights, our M-ary precomputation-based method is designed to optimize both memory consumption and computational efficiency in structured scalar multiplication tasks.

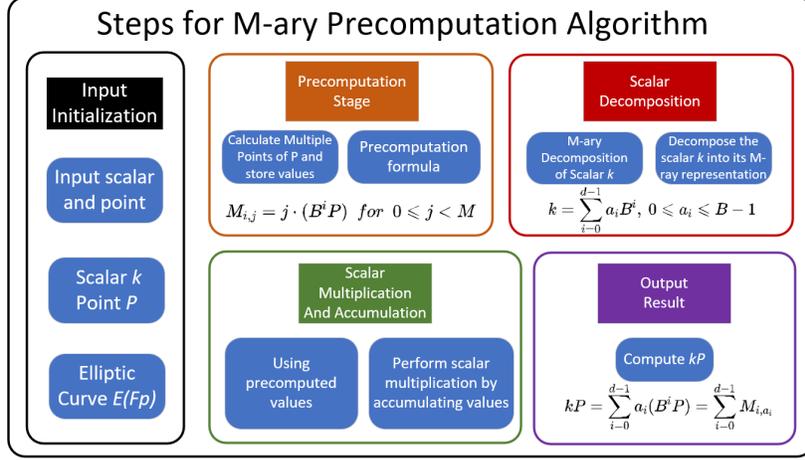


Figure 1: Steps for M-ary Precomputation-Based Algorithm, illustrating the process from input initialization, precomputation, scalar decomposition, scalar multiplication, and the final result computation.

Algorithm 3 M-ary Precomputation-Based Scalar Multiplication

Input: Scalars $k_1, \dots, k_Q \in \mathbb{N}$; base point $P \in E_p(a, b)$
Output: $(k_1P, k_2P, \dots, k_QP)$

// Step 1: Determine parameters d and B
 $d \leftarrow \left\lceil \frac{\ln p}{W(Q/e)+1} \right\rceil$
 $B \leftarrow \lceil \sqrt[d]{n} \rceil$
// Step 2: Precompute $M[i][j]$ for $i \in [0, d-1]$, $j \in [0, B]$
for $i = 0$ to $d-1$ **do**
 for $j = 0$ to B **do**
 Compute $M[i][j] \leftarrow j \cdot B^i \cdot P$
 Verify $M[i][j]$ satisfies the elliptic curve equation // *FI protection*
 end for
end for
// Step 3: For each scalar k , compute kP
for each $k \in \{k_1, \dots, k_Q\}$ **do**
 Choose random r and shift $k \leftarrow k + r \cdot B^d$
 Decompose k into base- B : $k = \sum_{i=0}^{d-1} k_i B^i$
 Compute $S \leftarrow \sum_{i=0}^{d-1} M[i][k_i]$
 Correct: $S \leftarrow S - r \cdot (B^d \cdot P)$
 Store S
end for
return all computed k_iP

Figure 1 provides a visual breakdown of each phase in the M-ary precomputation based algorithm. Additionally, Algorithm 3 offers a comprehensive explanation of the computational workflow for the M-ary precomputation-based accelerated scalar multiplication algorithms.

3.1. Overview of the Problem

Scalar multiplication on elliptic curves is a fundamental operation in many cryptographic protocols. Given a point P on an elliptic curve $E_p(a, b)$, the goal is to compute kP for a scalar k . Traditional methods for scalar multiplication suffer from time complexities proportional to $\Theta(Q \log p)$, where Q is the number of multiplications and p is the size of the finite field. This motivates the search for algorithms that can achieve lower time complexities.

3.2. The M-ary Precomputation Method

Our method leverages the fixed nature of both the elliptic curve and the base point G . Given that the same scalar multiplication is repeatedly computed, we utilize the M-ary precomputation technique to precompute and store multiples of the base point.

3.2.1. Scalar Representation

We represent the scalar k as a sum of powers of a base B :

$$k = \sum_{i=0}^{d-1} a_i B^i, \quad 0 \leq a_i \leq B - 1. \quad (18)$$

This allows the scalar multiplication to be expressed as:

$$kP = \sum_{i=0}^{d-1} a_i (B^i P), \quad (19)$$

where each term $B^i P$ is precomputed and stored in a table M , which significantly reduces the computational overhead when performing scalar multiplications.

3.2.2. Precomputing the Table M

The table M is defined as:

$$M_{i,j} = j (B^i P), \quad 0 \leq i \leq d-1, \quad 0 \leq j \leq B. \quad (20)$$

This table can be precomputed with a time complexity of $\Theta(dB)$. By using the recurrence relation:

$$M_{i,j} = M_{i,j-1} + M_{i,1}, \quad j \geq 2, \quad (21)$$

and the base case:

$$M_{i,1} = B^i P, \quad (22)$$

we can efficiently fill the table for all values of i and j . Once precomputation is done, the scalar multiplication kP can be quickly computed by summing the appropriate values from the table M :

$$kP = \sum_{i=0}^{d-1} M_{i,a_i}. \quad (23)$$

Thus, the time complexity for computing a single scalar multiplication is $\Theta(d)$, and for computing Q scalar multiplications, it becomes $\Theta(dQ)$.

3.3. Optimizing Time Complexity

To further optimize the algorithm, we seek to minimize the total computational cost by selecting the appropriate parameters for d and B . The relationship between B and d is derived from the requirement that $B^d - 1 \geq n - 1$, where n is the maximum possible value for the scalar k .

Since the time complexity of the algorithm is $\Theta(d(B + Q))$, we aim to minimize the expression by selecting B that balances both the base B and the number of precomputed values. Through mathematical analysis (see Appendix B for detailed proofs), the optimal value of B is found to be:

$$B^* = \lceil \sqrt[d]{n} \rceil. \quad (24)$$

Substituting this into the time complexity expression gives us:

$$\Theta(d(\sqrt[d]{n} + Q)) \quad (25)$$

3.4. Minimizing the Parameter d

To determine the optimal value of d , we seek the minimum of the function:

$$f(x) = x \left(p^{\frac{1}{x}} + Q \right), \quad x > 0. \quad (26)$$

By differentiating $f(x)$ and solving for the critical point, we find that the optimal value of d occurs at:

$$x_0 = \frac{\ln p}{W\left(\frac{Q}{e}\right) + 1}, \quad (27)$$

where $W(\cdot)$ is the principal branch of the Lambert W function. This value of d minimizes the overall time complexity, leading to an optimal trade-off between the precomputation cost and the number of scalar multiplications. Thus, $Q = O(f(x_0))$. Since $f(x_0) \leq f(d) < f(x_0) + \Theta(Q)$, we conclude:

$$f(d) = \Theta(f(x_0)) = \Theta\left(\frac{Q \ln p}{W\left(\frac{Q}{e}\right)}\right) = \Theta\left(\frac{Q \log p}{\log Q}\right). \quad (28)$$

3.5. Space Complexity

The spatial complexity arises from the precomputation of the table M , where $M_{i,j} = j(B^i P)$, for $0 \leq i \leq d-1$ and $0 \leq j \leq B$.

When $B = \lceil \sqrt[d]{n} \rceil$, $x_0 = \frac{\ln p}{W\left(\frac{Q}{e}\right) + 1}$, and $d = \lceil x_0 \rceil$, the spatial complexity is:

$$\Theta(dB) = \Theta\left(\frac{\ln p}{W\left(\frac{Q}{e}\right) + 1} \cdot p^{\frac{W\left(\frac{Q}{e}\right) + 1}{\ln p}}\right). \quad (29)$$

Simplifying the exponential term, we get:

$$\Theta\left(\frac{\ln p}{W\left(\frac{Q}{e}\right) + 1} \cdot \exp\left(W\left(\frac{Q}{e}\right) + 1\right)\right). \quad (30)$$

Since $\exp\left(W\left(\frac{Q}{e}\right) + 1\right) = e \cdot \frac{Q}{W\left(\frac{Q}{e}\right)}$, the spatial complexity becomes:

$$\Theta\left(\frac{\ln p}{W\left(\frac{Q}{e}\right) + 1} \cdot \frac{Q}{W\left(\frac{Q}{e}\right)}\right). \quad (31)$$

Thus, the final expression simplifies to:

$$\Theta\left(\frac{Q \log p}{\log^2 Q}\right). \quad (32)$$

3.6. The M -ary Precomputation Method with Less Space

When the number of input scalar multiplications Q is large, storing the precomputed points in table M with $\Theta\left(\frac{Q \log p}{\log^2 Q}\right)$ memory overhead incurs significant memory costs, which may diminish the practical advantages of our method. The challenge of managing precomputed data efficiently in scalar multiplication has been emphasized in recent works like Elastic MSM [39], which advocate elastic and modular preprocessing to adapt to memory constraints.

Building upon similar motivations, we design a binary sparse storage format for M -ary precomputation, significantly minimizing the number of stored points without compromising computational efficiency. In this approach, the points stored in each row are represented by a few 'indices' that span between powers of two. Specifically, we construct the storage format as follows:

$$\{M_{i,2^0}, M_{i,2^1}, M_{i,2^2}, \dots, M_{i,2^{\log_2 B}}\}, \quad 1 \leq i \leq d. \quad (33)$$

This method allows us to perform binary decomposition for each $M_{i,j}$, thus reducing the required storage space.

3.6.1. Bisection-Based Storage for Table M

We express a_i as the sum of powers of 2:

$$a_i = \sum_{j=0}^{\lfloor \log_2 B \rfloor} b_j 2^j, \quad kP = \sum_{i=0}^{d-1} \sum_{j=0}^{\lfloor \log_2 B \rfloor} b_j 2^j (B^i P). \quad (34)$$

First, we compute the complete precomputation table M using the conventional method, in which the total number of additions is approximately $O(d \cdot B)$, generating all elements of $M_{i,j}$. Subsequently, we select the bisection points from table M and store them in a compact table H :

$$H_{i,j} = M_{i,2^j}, \quad 1 \leq i \leq d, \quad 1 \leq j \leq \lfloor \log_2 B \rfloor. \quad (35)$$

We propose storing only table H , which enables efficient computation of kP by selectively retrieving and combining the appropriate precomputed points from H during scalar multiplication:

$$kP = \sum_{i=0}^{d-1} \sum_{j=0}^{\lfloor \log_2 B \rfloor} b_j H_{i,j}. \quad (36)$$

3.6.2. Determination of Parameter d

Since the complete table M must be computed during precomputation, the time complexity is $\Theta(dB)$. After adopting bisection-based storage, each digit a_i requires $\log_2 B$ scalar additions, resulting in a time complexity of $\Theta(d \log_2 B)$ for Q scalar multiplications. The overall time complexity is as follows:

$$\Theta(d(\sqrt[d]{p} + Q \log \sqrt[d]{p})). \quad (37)$$

By minimizing this time complexity, we can derive the optimal parameter d for the bisection-based storage scheme as:

$$d = \log_2 p, \quad B = \sqrt[d]{p} = e. \quad (38)$$

3.6.3. Complexity Analysis

We analyze the algorithm's complexity in terms of space and time requirements. Let d denote the decomposition depth and B the base used in the B -ary representation, with p being the bit-length of the field and Q the number of scalars.

Space Complexity. The algorithm maintains a precomputed table M of size $d \times B$, resulting in a space complexity of:

$$\Theta(dB) = \Theta(e \log p). \quad (39)$$

Time Complexity. For each scalar multiplication, the algorithm performs d additions, each involving a constant-time table lookup. For Q scalar multiplications, the total time complexity becomes:

$$\Theta(dQ \log B) = \Theta(Q \log p + e \log p) = \Theta(Q \log p). \quad (40)$$

Storage of Binary Elements. Each row stores a bounded number of intermediate points, typically logarithmic in B , leading to an auxiliary storage complexity of $\Theta(\log p)$.

Overall, the algorithm achieves near-linear time scaling in Q and logarithmic overhead in both precomputation and storage. Despite the reduced storage overhead, the binary storage scheme introduces additional computational cost during scalar multiplication, as reconstructing the required values from the compact table H involves more additive operations. Therefore, this optimization is particularly suitable for deployment on resource-constrained devices where memory is limited, but additional computation is acceptable during runtime.

3.7. Algorithm Security Analysis

This section analyzes the security properties of the proposed M-ary pre-computation based scalar multiplication algorithm from a theoretical perspective, focusing on its resistance to side-channel threats, fault injection vulnerabilities, and overall cryptographic soundness.

3.7.1. Side-Channel Resilience

To mitigate side-channel threats such as Simple Power Analysis (SPA) and Differential Power Analysis (DPA), the proposed algorithm incorporates several defensive mechanisms. First, all point additions are performed using a constant-time addition subroutine to prevent information leakage through timing variations. Second, a randomized scalar blinding technique is applied by transforming the scalar k to $k + r \cdot B^d$, introducing entropy across executions. Finally, table lookups during scalar decomposition follow a regular access pattern without branching, thereby avoiding conditional data-dependent operations.

These strategies collectively provide a baseline level of protection against timing-based and power-based leakage [40, 21]. However, for deployment in highly adversarial environments, further enhancements such as unified point addition formulas or dummy table accesses may be required to strengthen resistance against advanced power analysis.

3.7.2. Fault Injection Vulnerabilities

Fault injection (FI) attacks pose a different class of threats [41], where adversaries may attempt to manipulate internal states or induce faults in intermediate computations, particularly during table precomputation or scalar decomposition. To address this risk, the algorithm introduces a lightweight consistency check during the precomputation phase, where each point $M[i][j]$ is verified to satisfy the elliptic curve equation $y^2 = x^3 + ax + b$.

Despite this verification step, the algorithm currently lacks built-in redundancy or self-correction mechanisms. If a fault alters the value of a pre-computed point or disrupts scalar decomposition, the algorithm may output incorrect results without detection. Future work could explore fault-resilient enhancements such as result consistency checks, parity-based encoding, or lightweight checksum techniques that preserve performance while improving robustness against active attacks.

Table 3: Comparative Security Features of Scalar Multiplication Algorithms

Algorithm	SPA Resistance	DPA Resistance	FI Resistance
Double-and-Add	×	×	×
NAF-based	△	×	×
2^k -ary	△	×	×
Sliding Window	△	×	×
Montgomery Ladder	✓	△	×
Fixed-Base Comb	△	×	×
Pre-computation Scheme	△	×	×
M-ary (ours)	✓	△	△

✓: Strong built-in resistance △: Partial mitigation ×: No inherent protection

3.7.3. Theoretical Soundness

From a theoretical standpoint, the algorithm ensures correctness in scalar multiplication, preserving group law properties on elliptic curves. Under the assumption that elliptic curve operations are implemented securely, the algorithm maintains indistinguishability under chosen plaintext attacks (IND-CPA), which is the foundational requirement for elliptic curve-based encryption schemes.

However, practical instantiations may still leak information through side channels or fault manipulation unless complemented by secure hardware support or cryptographic protocol-level protections. For applications in digital signatures, secure multiparty computation, or zero-knowledge settings, additional safeguards such as verifiable computation or range proofs may be necessary.

Table 3 summarizes the resistance of several representative scalar multiplication algorithms against common cryptographic attacks, including Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Fault Injection (FI). As shown, our proposed M-ary algorithm achieves strong SPA resistance through constant-time computation and partial mitigation against DPA and FI via scalar randomization and lightweight consistency checks. Compared to classical methods such as double-and-add or sliding window, the proposed design offers improved baseline protection while maintaining lightweight efficiency.

4. Evaluation

In this section, we present an in-depth evaluation of the proposed M-ary precomputation-based accelerated scalar multiplication algorithm. The evaluation is conducted in three main phases:

First, we compare the theoretical algorithmic efficiency of our method against traditional scalar multiplication optimization algorithms, including Double-and-Add, NAF-based, and 2^k -ary algorithms. This comparison highlights the potential efficiency gains of the proposed method from a computational complexity perspective.

Second, we evaluate the performance of various scalar multiplication optimization algorithms within the ElGamal cryptosystem. By incorporating each algorithm into the ElGamal encryption scheme and performing scalar multiplications over elliptic curves `secp256k1`, `secp384r1`, and `secp521r1`, we measure improvements in encryption and decryption times. The results demonstrate that the M-ary precomputation-based algorithm significantly accelerates cryptographic operations compared to other methods. This analysis underscores the practical applicability of our proposed algorithm in real-world cryptographic systems, showcasing its effectiveness beyond theoretical models.

Lastly, we conduct a series of NS3-based simulations to evaluate the practical performance impact of the proposed algorithm within a representative communication network framework. The simulated environment consists of five interconnected nodes, and the evaluation focuses on key metrics such as total encryption time, entire communication time, and overall simulation time. The results demonstrate that the M-ary precomputation-based algorithm effectively reduces encryption time and improves communication efficiency within the simulated network environment.

These evaluations encompass theoretical efficiency analysis, practical simulation experiments, and cryptographic performance testing. Together, they provide a comprehensive validation of the advantages offered by the M-ary precomputation-based algorithm in the context of elliptic curve cryptography.

4.1. Evaluation on Theoretical Algorithmic Efficiency

In the previous section, we analyzed the complexities of various scalar multiplication algorithms, including the Double-and-Add, NAF-based, 2^k -ary, Sliding Window, Montgomery Ladder, Fixed-Base Comb, Fixed-Window,

Table 4: Comparative Analysis of Time and Space Complexities for Elliptic Curve Scalar Multiplication Algorithms.

Algorithm	Time Complexity	Space Complexity
Double-and-add	$\Theta(Q \log p)$	$\Theta(1)$
NAF-based	$\Theta(Q \log p)$	$\Theta(1)$
2^k -ary	$\Theta(Q \log p)$	$\Theta(1)$
Sliding Window	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$
Montgomery Ladder	$\Theta(Q \log p)$	$\Theta(1)$
Fixed-Base Comb	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$
Window τ -NAF	$\Theta\left(\frac{Q \log p}{r}\right)$	$\Theta(2^r)$
M-ary (ours)	$\Theta\left(\frac{Q \log p}{\log Q}\right)$	$\Theta\left(\frac{Q \log p}{\log^2 Q}\right)$
M-ary (binary)	$\Theta(Q \log p)$	$\Theta(\log p)$

and M-ary precomputation-based algorithms. As shown in Table 4, the M-ary algorithm achieves superior time complexity compared to conventional methods, which generally offer advantages in space complexity. In many practical cryptographic applications, space complexity remains within acceptable bounds, making time efficiency the primary criterion for evaluating the performance of scalar multiplication algorithms. Therefore, our M-ary precomputation-based approach is theoretically more effective for real-world deployment.

4.2. Evaluation on Quantitative Algorithmic Efficiency

4.2.1. Evaluation Setup

To investigate the performance comparison of the four scalar multiplication algorithms mentioned above, we have selected three elliptic curves (secp256k1, secp384r1, secp521r1) defined by the Standards for Efficient Cryptography Group for testing purposes. The parameter quintuples $T = (p, a, b, G, n)$ for each elliptic curve are provided in the Appendix.

4.2.2. Evaluation on the Efficiency of Scalar Multiplication Algorithms

This evaluation investigates the impact of the number of computations Q and base size p on the efficiency of four scalar multiplication algorithms for points on the elliptic curve $E_p(a, b)$. Quantitative results are presented in Figure 2, where Figures Figure 2a, Figure 2b, and Figure 2c illustrate the comparative efficiency across different base field sizes p .

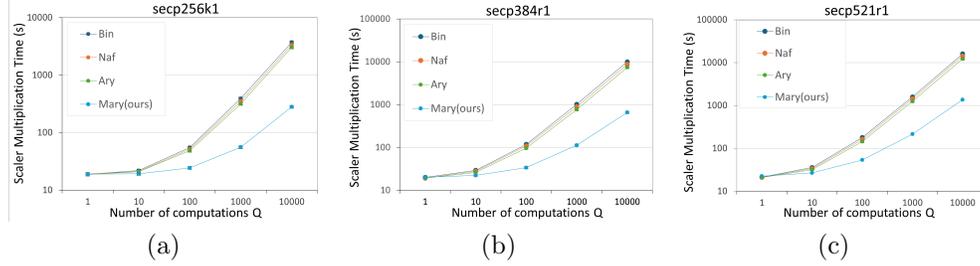


Figure 2: Evaluation on the time consumed by various optimized algorithms for scalar multiplication with increasing number of computations Q . Specifically, Figure 2a, Figure 2b, and Figure 2c represent the runtime for the elliptic curves secp256k1, secp384r1, and secp521r1, respectively.

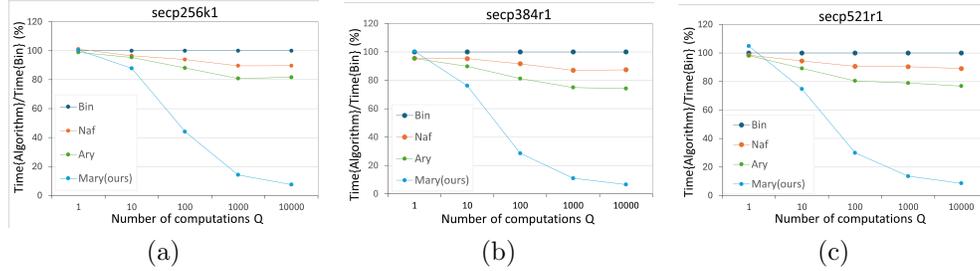


Figure 3: Evaluation on the proportion of time consumed by various optimized algorithms for scalar multiplication relative to the double-and-add algorithm as the number of computations Q increases. Specifically, Figure 3a, Figure 3b, and Figure 3c represent the proportion for the elliptic curves secp256k1, secp384r1, and secp521r1, respectively.

As Q increases, the efficiency advantage of our proposed algorithm over other optimized algorithms becomes more pronounced. Figure 3 shows that when Q is large, our algorithm consumes approximately 10% of the time required by the Double-and-Add algorithm, while the NAF-based algorithm takes about 90% and the 2^k -ary algorithm about 79%. This improvement is due to the time complexity of precomputation-based optimization algorithms, which includes a factor of $\frac{1}{\log Q}$; hence, as Q increases, the optimization effect intensifies. These experimental results align with our theoretical analysis.

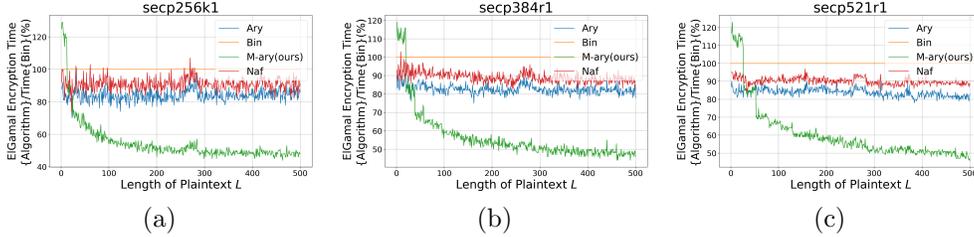


Figure 4: Evaluation of the proportion of time consumed by ElGamal Encryption using various optimized scalar multiplication algorithms compared to the double-and-add algorithm, as the plaintext length L increases. Specifically, Figure 4a, Figure 4b, and Figure 4c represent the respective proportions for the elliptic curves secp256k1, secp384r1, and secp521r1.

4.2.3. Evaluation on the Performance of Different Scalar Multiplication Optimization Algorithms in ElGamal Cryptosystem

We evaluate our M-ary Precomputation-Based Accelerated Scalar Multiplication Algorithm within the ElGamal cryptosystem, comparing its performance against traditional methods such as Double-and-Add, NAF-based, and 2^k -ary algorithms.

During encryption, User B computes $C_1 = kG$ and $C_2 = P_m + kP_A$, transmitting the ciphertext $C_m = (C_1, C_2)$ to User A. Since the base point G and public key P_A are fixed, our M-ary algorithm accelerates scalar multiplications for kG and kP_A . The algorithm’s advantages increase with the number of scalar multiplications Q .

To map a message m to a point on the elliptic curve E , we use a probabilistic mapping algorithm that encodes m into P_m . Each message is divided into groups of size k and represented as base-256 numbers mapped to points on E . Our algorithm computes all scalar multiplications simultaneously, significantly reducing encryption time.

Using elliptic curves secp256k1, secp384r1, and secp521r1, we compared the time for ElGamal encryption across various plaintext lengths L . As shown in Figure 4 and Table 5, our method outperforms traditional algorithms as L increases. Specifically, when L is large, the NAF and 2^k -ary algorithms require 78% to 92% of the time of the Double-and-Add method, while our approach only requires about 46%.

These results demonstrate the superior performance of our algorithm, particularly as the scalar size L increases. This improvement stems from the

Table 5: Comparative Analysis of Time Consumed by Different Scalar Multiplication Algorithms used in ElGamal encryption when Length of Plaintext $L = 100, 300, 360$.

Algorithm	Curve	Time Consumed (seconds)		
		$L = 100$	$L = 300$	$L = 360$
Double-and-add	secp256k1	0.18s	0.49s	0.79s
	secp384r1	0.26s	0.71s	1.16s
	secp521r1	0.36s	1.10s	1.83s
NAF-based	secp256k1	0.16s	0.44s	0.72s
	secp384r1	0.24s	0.62s	0.95s
	secp521r1	0.33s	0.96s	1.61s
2^k -ary	secp256k1	0.14s	0.40s	0.68s
	secp384r1	0.22s	0.59s	0.91s
	secp521r1	0.30s	0.89s	1.51s
M-ary Precomputation (ours)	secp256k1	0.10s	0.24s	0.38s
	secp384r1	0.15s	0.35s	0.54s
	secp521r1	0.24s	0.57s	0.85s

time complexity factor $\frac{1}{\log Q}$ in our method, which ensures enhanced efficiency as Q scales with L . The empirical findings validate the theoretical analysis and highlight the strong scalability of our approach, making it well-suited for high-performance cryptographic applications. These results provide a solid foundation for further evaluation in broader communication and security contexts.

4.3. Evaluation in Communication-Oriented Simulation Using NS3 Framework

To further assess the practical performance of the proposed scalar multiplication algorithms, we conducted a series of simulations using the NS3 framework. These experiments were designed to evaluate the algorithms within a representative communication network setting, focusing on key metrics such as encryption time, communication time, and overall simulation time.

4.3.1. Simulation Setup

The simulation environment modeled a communication network with five interconnected nodes, each representing a device performing secure data transmission. As illustrated in Figure 5a, the nodes were connected using point-to-point channels with a bandwidth of 5 Mbps and a delay of 2 ms, reflecting typical parameters for lightweight communication systems.

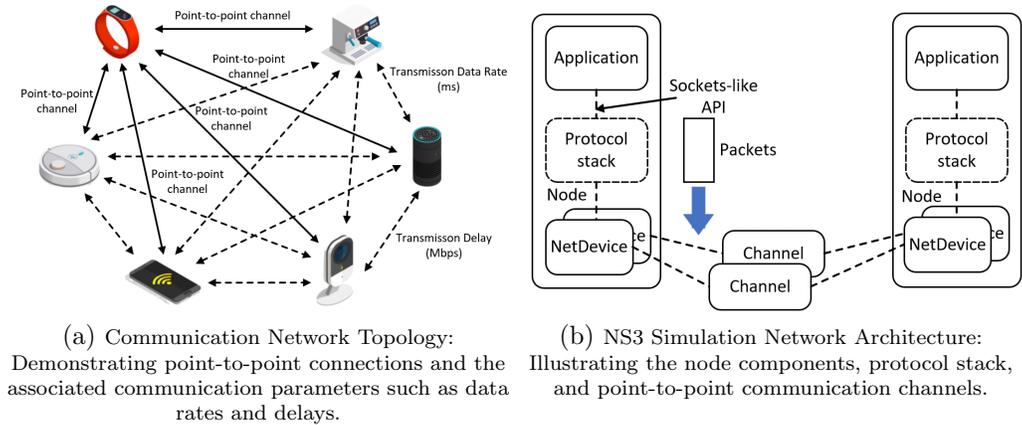


Figure 5: NS3 simulation communication model. Figure 5a demonstrates the overall network topology with point-to-point links and their associated communication metrics, such as transmission delays and data rates. Figure 5b illustrates the internal architecture of nodes in the NS3 simulation, showing the application layer, protocol stack, and net device components.

To evaluate the performance of different scalar multiplication algorithms, we encrypted a set of predefined text messages of varying sizes, simulating practical data transmission scenarios. The elliptic curve-based ElGamal encryption scheme was employed, incorporating eight scalar multiplication algorithms: Double-and-Add, NAF-based, 2^k -ary, Sliding Window, Montgomery Ladder, Fixed-Base Comb, Fixed-Window, and our proposed M-ary precomputation-based algorithm, including both the standard and binary-optimized versions.

Three primary performance indicators were recorded during the simulations:

- **Total Encryption Time (Enc. Time):** The total time required to encrypt all messages using the selected scalar multiplication algorithm.
- **Total Communication Time (Com. Time):** The overall duration required to transmit all encrypted messages across the network.
- **Total Simulation Time (Sim. Time):** The end-to-end duration of the complete simulation process, including encryption, transmission, and protocol handling.

Messages were encrypted prior to transmission using the selected algorithm, and the corresponding timing results were logged for comprehensive

evaluation. Each node’s internal structure, depicted in Figure 5b, comprises an application layer, a protocol stack, and network devices, reflecting a typical layered communication architecture.

4.3.2. Simulation Results

Table 6: Simulation Results for Different Scalar Multiplication Algorithms and Elliptic Curves.

Elliptic Curve	Algorithm	Enc. Time (s)	Com. Time (s)	Sim. Time (s)
secp256k1	Double-and-add	6.8971	6.8978	6.9041
	NAF-based	6.8802	6.8810	6.8909
	2^k -ary	6.8032	6.8039	6.8115
	Sliding Window	6.6316	6.6324	6.6397
	Montgomery Ladder	7.1261	7.1268	7.1360
	Fixed-Base Comb	6.7629	6.7636	6.7729
	Fixed-Window	6.8547	6.8554	6.8618
	M-ary (ours)	6.3361	6.3369	6.3491
	M-ary (binary)	6.61878	6.6197	6.6279
secp384r1	Double-and-add	8.5375	8.5382	8.5444
	NAF-based	8.3399	8.3407	8.3483
	2^k -ary	7.7314	7.7321	7.7385
	Sliding Window	7.9080	7.9087	7.9150
	Montgomery Ladder	9.2238	9.2245	9.2338
	Fixed-Base Comb	7.4359	7.4366	7.4431
	Fixed-Window	8.2275	8.2282	8.2345
	M-ary (ours)	7.0521	7.0528	7.0602
	M-ary (binary)	7.2438	7.2445	7.25407
secp521r1	Double-and-add	10.9273	10.9281	10.9376
	NAF-based	10.6670	10.6679	10.7002
	2^k -ary	9.7595	9.7603	9.7699
	Sliding Window	9.6087	9.6094	9.6160
	Montgomery Ladder	12.2517	12.2525	12.2626
	Fixed-Base Comb	11.8115	11.8123	11.8227
	Fixed-Window	10.4312	10.4319	10.4416
	M-ary (ours)	8.3926	8.3934	8.4002
	M-ary (binary)	8.5293	8.5300	8.5404

Time Efficiency Analysis. The NS3 simulation results demonstrate the enhanced efficiency of our proposed M-ary precomputation-based algorithm across different elliptic curve settings. Table 6 summarizes the encryption time, communication time, and overall simulation time for each algorithm across three elliptic curves: secp256k1, secp384r1, and secp521r1. Specifically, our method consistently achieves lower encryption times compared to other algorithms, leading to reductions in overall communication and simulation times, and highlighting its scalability for large-scale cryptographic applications.

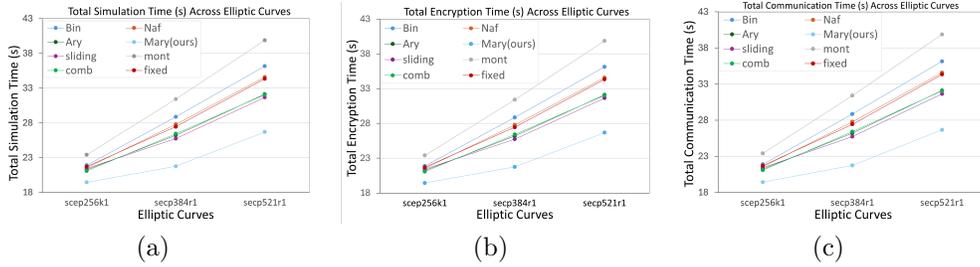


Figure 6: Performance Comparison of Scalar Multiplication Algorithms on Elliptic Curves. Figure 6a shows the total simulation time across three elliptic curves (secp256k1, secp384r1, secp521r1). Figure 6b depicts the total encryption time. Figure 6c presents the total communication time.

Figures 6a, 6b, and 6c provide a visual comparison of the performance metrics:

- Figure 6a shows the reduction in encryption time across multiple transmission sessions, highlighting the computational efficiency of the proposed method.
- Figure 6b depicts communication time savings, demonstrating the improved transmission performance achieved by our approach.
- Figure 6c illustrates that our algorithm consistently outperforms others in terms of total simulation time across all tested elliptic curves, showcasing its strong scalability with increasing computational complexity.

These results underscore the superior performance and practicality of our M-ary precomputation-based algorithm, making it highly suitable for high-performance and large-scale cryptographic applications that require both speed and computational efficiency.

Memory Consumption Analysis. Peak memory usage is one of the most critical metrics for evaluating the deployability and scalability of scalar multiplication algorithms. Rather than only focusing on the theoretical size of precomputation tables, we measure the *peak memory usage* during the entire execution process, including temporary variables, intermediate buffers, and any auxiliary structures used during computation. This better reflects the algorithm’s actual memory footprint in practical systems.

To systematically compare the memory performance of different algorithms, we selected three standard elliptic curves—secp256k1, secp384r1, and

secp521r1—and measured the peak memory usage for each of nine scalar multiplication algorithms under four scalar counts $Q \in \{1, 10, 100, 1000\}$. All measurements reflect the actual peak memory usage during execution, not just theoretical memory requirements. A comprehensive summary of the experimental results is provided in Table 7.

Table 7: Peak memory usage (in MB) of different scalar multiplication algorithms across three curves and four scalar counts $Q \in \{1, 10, 100, 1000\}$. Values in bold represent the two algorithms achieving the lowest peak memory consumption in each configuration.

Elliptic Curve	Algorithm	$Q = 1$	$Q = 10$	$Q = 100$	$Q = 1000$
secp256k1	Double-and-add	0.000969	0.002602	0.018410	0.177040
	NAF-based	0.000931	0.002560	0.018368	0.177002
	2^k -ary	0.003941	0.005093	0.016094	0.126659
	Sliding Window	0.032118	0.033269	0.044271	0.165089
	Montgomery Ladder	0.001081	0.002232	0.013234	0.123797
	Fixed-Base Comb	0.001528	0.003160	0.018969	0.161898
	Fixed-Window	0.007359	0.008991	0.020153	0.130718
	M-ary (ours)	0.000797	0.002430	0.018238	0.144852
	M-ary (binary)	0.000690	0.001842	0.012844	0.123409
secp384r1	Double-and-add	0.001152	0.003059	0.021614	0.207710
	NAF-based	0.001129	0.003036	0.021591	0.207687
	2^k -ary	0.004871	0.006298	0.020046	0.158077
	Sliding Window	0.040190	0.041616	0.055364	0.201939
	Montgomery Ladder	0.001370	0.002797	0.016544	0.154574
	Fixed-Base Comb	0.001787	0.003695	0.022248	0.192642
	Fixed-Window	0.009449	0.011356	0.025265	0.163296
	M-ary (ours)	0.000950	0.002857	0.021412	0.186707
	M-ary (binary)	0.000843	0.002270	0.016018	0.154049
secp521r1	Double-and-add	0.001377	0.003628	0.025616	0.246044
	NAF-based	0.001389	0.003635	0.025627	0.246052
	2^k -ary	0.005772	0.007542	0.024723	0.197086
	Sliding Window	0.050177	0.051948	0.069129	0.248326
	Montgomery Ladder	0.001706	0.003477	0.020658	0.193020
	Fixed-Base Comb	0.002146	0.004439	0.026430	0.231155
	Fixed-Window	0.011677	0.014477	0.031818	0.203709
	M-ary (ours)	0.001152	0.003403	0.025391	0.236874
	M-ary (binary)	0.001041	0.002815	0.019997	0.192360

Traditional algorithms such as Double-and-Add and NAF-based exhibit moderate memory usage overall. While they do not involve precomputation, their peak memory is not necessarily minimal. For instance, on secp384r1 with $Q = 100$, NAF-based consumes 0.0216 MB—higher than both Montgomery Ladder (0.0165 MB) and our M-ary (binary) method (0.0160 MB).

Fixed-Base Comb and Fixed-Window algorithms tend to show higher memory consumption, especially in small-scale tasks. For example, on secp256k1 with $Q = 10$, Fixed-Window uses 0.0090 MB, whereas our M-ary (binary) method requires only 0.0018 MB—almost 5 times smaller.

The Sliding Window algorithm, despite being configured with the minimal window size in our experiments, still incurs significant memory overhead due to its exponential precomputation structure. On secp521r1 with $Q =$

1, it reaches a peak of 0.0502 MB, while M-ary (binary) only uses 0.0010 MB—demonstrating a more than 50-fold reduction.

Montgomery Ladder, while not a precomputation-based method, achieves consistently low memory usage due to its simple structure without extra caching. However, it still falls short of our optimized binary method in most settings. Our original M-ary algorithm demonstrates stable performance across tasks, achieving lower memory usage than most traditional methods while maintaining computational efficiency.

Our M-ary (binary) algorithm consistently achieves the best memory efficiency. It ranks first in 9 out of 12 test cases, and top-two in all remaining cases. For example, on `secp256k1` with $Q = 10$, it uses just 0.0018 MB—lower than all other compared methods. On `secp384r1` with $Q = 100$, it consumes only 0.0160 MB (the lowest among all). On `secp521r1`, it ranks among the lowest in every Q setting. This optimization benefits from a sparse power representation, storing only 2^j multiples in each layer and reconstructing the target value through dynamic additions, thereby significantly reducing memory redundancy.

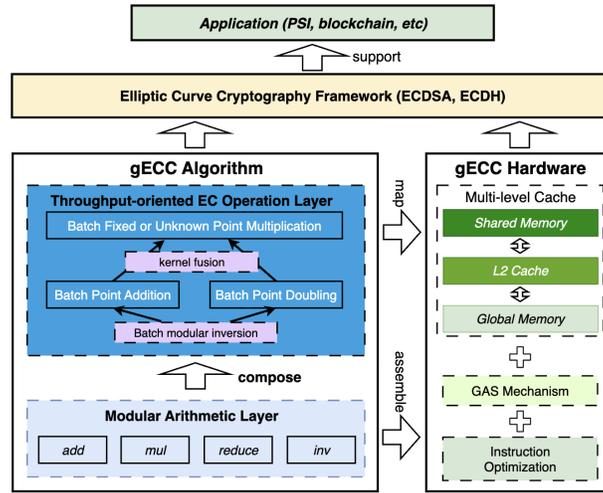
5. Potential for Hardware Implementation

Although this paper mainly focuses on theoretical analysis, the proposed M-ary precomputation-based scalar multiplication algorithm shows strong potential for hardware acceleration on platforms such as GPUs and FPGAs.

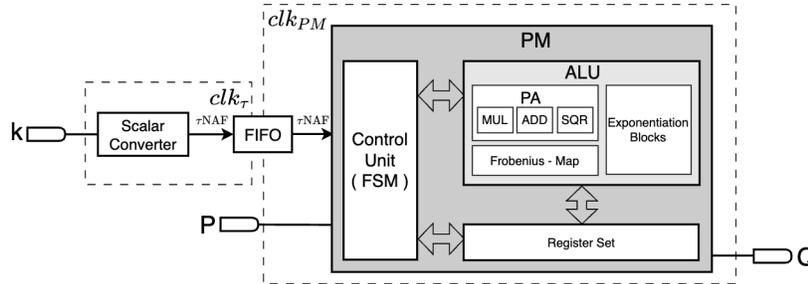
Recent studies, such as the gECC framework proposed by Xiong et al. [42], have demonstrated that optimizing batch scalar multiplications on GPUs can significantly improve the throughput of elliptic curve cryptographic (ECC) operations. Their framework, illustrated in Figure 7a, leverages Montgomery’s trick and efficient scheduling strategies to reduce memory access overhead, achieving high performance on NVIDIA A100 GPUs. Given the reduced time and memory complexity of our M-ary method, it is well-suited for integration into such parallel architectures, promising further improvements in latency and throughput.

On FPGA platforms, Marzouqi et al. [43] presented a high-speed ECC processor based on redundant signed digit (RSD) representations and pipelined Karatsuba–Ofman multipliers. Their results highlight that optimized scalar decompositions can effectively reduce scalar multiplication latency and area, which aligns closely with the objectives of our M-ary method.

Moreover, Jiang et al. [44] proposed low-latency and area-efficient point multiplication architectures over Koblitz curves. Their overall hardware architecture, shown in Figure 7b, optimizes scalar conversion and computation scheduling within pipelined designs to achieve significant latency reductions. Although their work specifically targets Koblitz curves, the core design principles—precomputation strategies, pipelined scheduling, and area-time trade-off optimization—are broadly applicable to general ECC scalar multiplication problems.



(a) gECC Framework for GPU-based Batch ECC Acceleration



(b) FPGA Architecture for Koblitz Curve Point Multiplication

Figure 7: Hardware Deployment Potentials of the proposed M-ary scalar multiplication method: (a) GPU-based high-throughput ECC framework; (b) FPGA-based scalar conversion and point multiplication architecture.

Therefore, the proposed M-ary algorithm not only improves theoretical ef-

efficiency but also exhibits strong potential for hardware compatibility. Future work will explore implementing the algorithm on modern GPU and FPGA platforms to validate its practical performance gains.

6. Conclusion

This paper presents an M-ary precomputation-based accelerated scalar multiplication algorithm for elliptic curve cryptography (ECC), designed to enhance computational efficiency and optimize memory usage. By leveraging structured precomputation and scalar decomposition, the proposed method reduces the time complexity from $\Theta(Q \log p)$ to $\Theta\left(\frac{Q \log p}{\log Q}\right)$ and achieves a flexible memory complexity of $\Theta\left(\frac{Q \log p}{\log^2 Q}\right)$, enabling efficient operation across a wide range of computational scales.

Comprehensive evaluations validate the effectiveness of the proposed approach. Theoretical analysis confirms its superior asymptotic efficiency compared to traditional scalar multiplication methods. In cryptographic experiments using the ElGamal encryption scheme, the proposed method achieves up to a 59% reduction in encryption time on the secp256k1 curve when $Q = 1000$, compared to Double-and-Add, NAF-based, and 2^k -ary algorithms. In NS3-based simulations, the M-ary (binary) variant achieves a 22% reduction in total communication time on the secp384r1 curve and up to a 25% reduction in overall simulation time on the secp521r1 curve. Memory performance is also significantly improved: for instance, on secp256k1 with $Q = 1000$, the M-ary (binary) method reduces peak memory usage by approximately 25% compared to the Sliding Window algorithm.

These results collectively underscore the versatility, scalability, and practical applicability of the M-ary precomputation-based scalar multiplication algorithm. The method not only enhances cryptographic performance but also provides a promising foundation for deployment in secure communication systems, large-scale cryptographic computations, and hardware-accelerated cryptographic applications.

7. Future Work

This work primarily focuses on optimizing scalar multiplication in single-scalar scenarios with a fixed base point. Several promising directions for future research are identified:

- **Extension to Multi-Scalar Multiplication (MSM):** Extending the proposed M-ary precomputation method to multi-scalar multiplication, which plays a critical role in applications such as pairing-based cryptography and signature aggregation, is a natural progression. Investigating dynamic parameter selection strategies for efficient multi-scalar operations remains an open challenge.
- **Parallelization and Batch Processing:** Exploring parallel and batch execution of the algorithm could further enhance its computational efficiency, particularly in settings where large volumes of scalar multiplications must be processed simultaneously.
- **Adaptation to Dynamic Curve Parameters and Base Points:** While this work assumes fixed elliptic curve parameters and base points, many real-world cryptographic systems involve variable curves or changing base points. Future research may explore incremental table update mechanisms to efficiently adapt precomputation structures without incurring significant recomputation overhead.
- **Hardware-Optimized Implementations:** Investigating optimized deployments of the algorithm on diverse hardware architectures, such as ARM, RISC-V, and GPU platforms, could further demonstrate its practical scalability and performance across different computational environments.

These directions offer valuable opportunities to further enhance the applicability, efficiency, and robustness of the proposed M-ary precomputation-based scalar multiplication method across a broad range of cryptographic applications.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62162057, No. 61872254), the Key Lab of Information Network Security of the Ministry of Public Security, China (C20606), the Sichuan Science and Technology Program, China (2021JDRC0004), and the Key Laboratory of Data Protection and Intelligent Management of the Ministry of Education, China (SCUSAKFKT202402Y). We express our gratitude to the corresponding authors, Tongxi Wu and Xufeng Liu, for their equal contributions and invaluable advice throughout the writing process of this paper.

Appendix A. Parameters of the Selected Elliptic Curves for Evaluation

We selected three elliptic curves defined by the Standards for Efficient Cryptography Group (SECG) for evaluation: secp256k1, secp384r1, and secp521r1. Each elliptic curve is defined by a quintuple $T = (p, a, b, G, n)$. The parameters are as follows (all represented in hexadecimal):

Appendix A.1. secp256k1

The finite field \mathbb{F}_p of secp256k1 is defined by the prime p :

$$\begin{aligned} p &= \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFF FFFFC2F} \\ &= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \end{aligned}$$

The elliptic curve E over \mathbb{F}_p is defined by the Weierstrass equation:

$$E : y^2 = x^3 + ax + b,$$

where

$$\begin{aligned} a &= \text{00000000 00000000 00000000} \\ &\quad \text{00000000 00000000 00000000} \\ &\quad \text{00000000 00000000} \\ b &= \text{00000000 00000000 00000000} \\ &\quad \text{00000000 00000000 00000000} \\ &\quad \text{00000000 00000007} \end{aligned}$$

The base point $G = (x, y)$ on the curve is defined as:

$$\begin{aligned} x &= \text{79BE667E F9DCBBAC 55A06295} \\ &\quad \text{CE870B07 029BFCDB 2DCE28D9} \\ &\quad \text{59F2815B 16F81798} \\ y &= \text{483ADA77 26A3C465 5DA4FBFC} \\ &\quad \text{0E1108A8 FD17B448 A6855419} \\ &\quad \text{9C47D08F FB10D4B8} \end{aligned}$$

The order n of the base point G is:

$$\begin{aligned}n &= \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFF BAAEDCE6 AF48A03B} \\ &\quad \text{BFD25E8C D0364141}\end{aligned}$$

Appendix A.2. secp384r1

The finite field \mathbb{F}_p of secp384r1 is defined as:

$$\begin{aligned}p &= \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{00000000 00000000 FFFFFFFF} \\ &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1\end{aligned}$$

The elliptic curve E over \mathbb{F}_p is defined by the Weierstrass equation $y^2 = x^3 + ax + b$, where a and b are:

$$\begin{aligned}a &= \text{FFFFFFFF FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFE FFFFFFFF FFFFFFFF} \\ &\quad \text{FFFFFFFE FFFFFFFE FFFFFFFF} \\ &\quad \text{00000000 00000000 FFFFFFFC} \\ b &= \text{B3312FA7 E23EE7E4 988E056B} \\ &\quad \text{E3F82D19 181D9C6E FE814112} \\ &\quad \text{0314088F 5013875A C656398D} \\ &\quad \text{8A2ED19D 2A85C8ED D3EC2AEF}\end{aligned}$$

The base point $G = (x, y)$ has coordinates:

$$\begin{aligned}x &= \text{AA87CA22 BE8B0537 8EB1C71E} \\ &\quad \text{F320AD74 6E1D3B62 8BA79B98} \\ &\quad \text{59F741E0 82542A38 5502F25D} \\ &\quad \text{BF55296C 3A545E38 72760AB7} \\ y &= \text{3617DE4A 96262C6F 5D9E98BF} \\ &\quad \text{9292DC29 F8F41DBD 289A147C} \\ &\quad \text{E9DA3113 B5F0B8C0 0A60B1CE} \\ &\quad \text{1D7E819D 7A431D7C 90EA0E5F}\end{aligned}$$

The order n of the base point G is:

$$\begin{aligned}n &= \text{FFFFFFFF FFFFFFFFF FFFFFFFFF} \\ &\quad \text{FFFFFFFF FFFFFFFFF FFFFFFFFF} \\ &\quad \text{C7634D81 F4372DDF 581A0DB2} \\ &\quad \text{48B0A77A ECEC196A CCC52973}\end{aligned}$$

Appendix A.3. secp521r1

The finite field \mathbb{F}_p of secp521r1 is defined as:

$$\begin{aligned}p &= \text{01FF FFFFFFFFF FFFFFFFFF} \\ &\quad \text{FFFFFFFF FFFFFFFFF} \\ &= 2^{521} - 1\end{aligned}$$

The elliptic curve E over \mathbb{F}_p is defined by the Weierstrass equation $y^2 =$

$x^3 + ax + b$, where a and b are:

```
a = 01FF FFFFFFFF FFFFFFFF
   FFFFFFFF FFFFFFFC
b = 0051 953EB961 8E1C9A1F
   929A21A0 B68540EE A2DA725B
   99B315F3 B8B48991 8EF109E1
   56193951 EC7E937B 1652C0BD
   3BB1BF07 3573DF88 3D2C34F1
   EF451FD4 6B503F00
```

The base point $G = (x, y)$ has coordinates:

```
x = 00C6 858E06B7 0404E9CD
   9E3ECB66 2395B442 9C648139
   053FB521 F828AF60 6B4D3DBA
   A14B5E77 EFE75928 FE1DC127
   A2FFA8DE 3348B3C1 856A429B
   F97E7E31 C2E5BD66
y = 0118 39296A78 9A3BC004
   5C8A5FB4 2C7D1BD9 98F54449
   579B4468 17AFBD17 273E662C
   97EE7299 5EF42640 C550B901
   3FAD0761 353C7086 A272C240
   88BE9476 9FD16650
```

The order n of the base point G is:

```
n = 01FF FFFFFFFF FFFFFFFF
    FFFFFFFF FFFFFFFF FFFFFFFF
    FFFFFFFF FFFFFFFF FFFFFFFF
    51868783 BF2F966B 7FCC0148
    F709A5D0 3BB5C9B8 899C47AE
    BB6FB71E 91386409
```

Appendix B. Mathematical Proofs and Theorems

Appendix B.1. Proof of NAF Representation and Its Properties

This section presents theorems and proofs foundational to the NAF-based scalar multiplication algorithm.

Theorem 1. An integer n can be represented in signed binary form as a sequence $a_{d-1}, a_{d-2}, \dots, a_0$, satisfying:

$$n = \sum_{i=0}^{d-1} a_i \cdot 2^i, \quad (\text{B.1})$$

where $a_i \in \{-1, 0, 1\}$ and $a_{d-1} \neq 0$.

Theorem 2. The number of non-zero elements in the signed binary representation of n is the Hamming weight, denoted as:

$$\sum_{i=0}^{d-1} [a_i \neq 0]. \quad (\text{B.2})$$

Theorem 3. The signed binary representation of n is called the Non-Adjacent Form (NAF) if $a_i a_{i+1} = 0$ for all $0 \leq i < d - 1$. It is denoted as $(a_{d-1} a_{d-2} \dots a_1 a_0)_{\text{NAF}}$.

Theorem 4. The NAF of a non-negative integer n is unique and denoted as $\text{NAF}(n)$.

Theorem 5. The Hamming weight of $\text{NAF}(n)$ is the smallest among all signed binary representations of n .

Theorem 6. The length $\ell(n)$ of the NAF of a positive integer n satisfies:

$$\lfloor \log_2 n \rfloor + 1 \leq \ell(n) \leq \lceil \log_2 n \rceil + 1. \quad (\text{B.3})$$

These theorems form the mathematical basis for understanding the Non-Adjacent Form and its application in scalar multiplication.

Appendix B.2. Theorem 7: Integer Representation in Base B

Theorem 7. For a positive integer $B \geq 2$ and $n \in \mathbb{N}$, there exist $d \in \mathbb{N}$ and $a_0, a_1, \dots, a_{d-1} \in [0, B - 1]$ such that:

$$n = \sum_{i=0}^{d-1} a_i B^i. \quad (\text{B.4})$$

Proof: Let $d := \lceil \log_B(n + 1) \rceil$. Then $d \in \mathbb{N}$, and for $0 \leq i \leq d - 1$, $0 \leq a_i < B$ holds.

$$n = \sum_{i=0}^{d-1} a_i B^i = \sum_{i=0}^{d-1} \left(\left\lfloor \frac{n}{B^i} \right\rfloor \bmod B \right) B^i. \quad (\text{B.5})$$

This simplifies to:

$$n = \sum_{i=0}^{d-1} \left(\left\lfloor \frac{n}{B^i} \right\rfloor - \left\lfloor \frac{n}{B^{i+1}} \right\rfloor B \right) B^i. \quad (\text{B.6})$$

Rearranging gives:

$$n = \left\lfloor \frac{n}{B^0} \right\rfloor B^0 - \left\lfloor \frac{n}{B^d} \right\rfloor B^d. \quad (\text{B.7})$$

Since $B^d \geq n + 1 > n$, $\left\lfloor \frac{n}{B^d} \right\rfloor = 0$, hence:

$$n = \left\lfloor \frac{n}{B^0} \right\rfloor B^0 = n. \quad (\text{B.8})$$

Appendix B.3. Scalar Multiplication Using Theorem 7

To compute scalar multiplication kP , we represent k as:

$$k = \sum_{i=0}^{d-1} a_i B^i, \quad 0 \leq a_i < B. \quad (\text{B.9})$$

Thus, scalar multiplication can be expressed as:

$$kP = \sum_{i=0}^{d-1} a_i (B^i P). \quad (\text{B.10})$$

We precompute an array M defined as:

$$M_{i,j} = j (B^i P), \quad 0 \leq i < d, \quad 0 \leq j < B. \quad (\text{B.11})$$

For $j \geq 2$, the array M satisfies:

$$M_{i,j} = M_{i,j-1} + M_{i,1}. \quad (\text{B.12})$$

For $j = 1$:

$$M_{i,1} = B^i P = \begin{cases} M_{i-1,B}, & \text{if } i \geq 1, \\ P, & \text{if } i = 0. \end{cases} \quad (\text{B.13})$$

For $j = 0$:

$$M_{i,0} = O. \quad (\text{B.14})$$

The array M can be precomputed with a time complexity of $\Theta(dB)$.

After precomputing M , the scalar multiplication kP can be expressed as:

$$kP = \sum_{i=0}^{d-1} a_i (B^i P) = \sum_{i=0}^{d-1} M_{i,a_i}. \quad (\text{B.15})$$

Thus, the time complexity for a single scalar multiplication is $\Theta(d)$, and for Q scalar multiplications, it is $\Theta(dQ)$. Including the precomputation time, the total complexity is:

$$\Theta(d(B + Q)). \quad (\text{B.16})$$

Appendix B.4. Minimization of Time Complexity

Since the coefficient k could be as high as $n - 1$, the parameters B and d must satisfy:

$$B^d - 1 \geq n - 1. \quad (\text{B.17})$$

To minimize $d(B + Q)$ with respect to B , we choose $B^* := \lceil \sqrt[d]{n} \rceil$. This leads to the overall time complexity:

$$\Theta(d(B + Q)) = \Theta(d(\sqrt[d]{n} + Q)) = \Theta(d(\sqrt[d]{p} + Q)). \quad (\text{B.18})$$

The remaining variable is d . To minimize:

$$f(x) = x \left(p^{\frac{1}{x}} + Q \right), \quad x > 0, \quad (\text{B.19})$$

we find the derivative $f'(x) = 0$:

$$x_0 = \frac{\ln p}{W\left(\frac{Q}{e}\right) + 1}, \quad (\text{B.20})$$

where $W(\cdot)$ is the Lambert W function's principal branch. This indicates $f(x)$ reaches its minimum at $x = x_0$.

Since f' is strictly increasing on $(0, +\infty)$, $f(x)$ is strictly decreasing on $(0, x_0]$ and strictly increasing on $[x_0, +\infty)$. Thus, $f(x)$ attains its minimum at $x = x_0$.

References

- [1] S. Das, S. Duan, S. Liu, A. Momose, L. Ren, V. Shoup, Asynchronous consensus without trusted setup or public-key cryptography, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 3242–3256.
- [2] S. Dolev, B. Guo, J. Niu, Z. Wang, Sodsbc: a post-quantum by design asynchronous blockchain framework, IEEE Transactions on Dependable and Secure Computing 21 (1) (2023) 47–62.
- [3] A. Bandarupalli, A. Bhat, S. Bagchi, A. Kate, M. K. Reiter, Random beacons in monte carlo: Efficient asynchronous random beacon without threshold cryptography, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 2621–2635.

- [4] S. Duan, X. Wang, H. Zhang, Fin: Practical signature-free asynchronous common subset in constant time, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 815–829.
- [5] Z. Cao, L. Liu, The practical advantage of rsa over ecc and pairings, Cryptology ePrint Archive (2024).
- [6] V. Dahiphale, H. Raut, G. Bansod, D. Dahiphale, Securing iot devices with fast and energy efficient implementation of pride and present ciphers, Cyber Security and Applications 3 (2025) 100055.
- [7] S. Ullah, J. Zheng, N. Din, M. T. Hussain, F. Ullah, M. Yousaf, Elliptic curve cryptography; applications, challenges, recent advances, and future trends: A comprehensive survey, Computer Science Review 47 (2023) 100530.
- [8] W. Haddaji, L. Ghammam, N. El Mrabet, L. B. Abdelghani, On computing the multidimensional scalar multiplication on elliptic curves, Cryptology ePrint Archive (2024).
- [9] A. Kumar, H. Om, Design of a usim and ecc based handover authentication scheme for 5g-wlan heterogeneous networks, Digital Communications and Networks 6 (3) (2020) 341–353.
- [10] S. Xie, S. Ma, M. Ding, Y. Shi, M. Tang, Y. Wu, Robust information bottleneck for task-oriented communication with digital modulation, IEEE Journal on Selected Areas in Communications 41 (8) (2023) 2577–2591.
- [11] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, E. Wustrow, Elliptic curve cryptography in practice, in: Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18, Springer, 2014, pp. 157–175.
- [12] R. Avanzi, C. Heuberger, H. Prodinger, Redundant τ -adic expansions i: non-adjacent digit sets and their applications to scalar multiplication, Designs, Codes and Cryptography 58 (2011) 173–202.
- [13] W. Zhang, Y. Liu, M. Meng, J. Lv, J. Wang, Y. Liu, Data encryption transmission method of communication network based on rsa algorithm,

- in: International Conference on Signal Processing and Communication Technology (SPCT 2022), Vol. 12615, SPIE, 2023, pp. 359–363.
- [14] M. Rivain, Fast and regular algorithms for scalar multiplication over elliptic curves, Cryptology ePrint Archive (2011).
 - [15] W. Yang, C. Hou, Y. Wang, Z. Zhang, X. Wang, Y. Cao, Sakms: A secure authentication and key management scheme for ietf 6tisch industrial wireless networks based on improved elliptic-curve cryptography, IEEE Transactions on Network Science and Engineering (2024).
 - [16] B. Ansari, M. A. Hasan, High-performance architecture of elliptic curve scalar multiplication, IEEE Transactions on Computers 57 (11) (2008) 1443–1453.
 - [17] N. A. Mohamed, M. H. Hashim, M. Hutter, Improved fixed-base comb method for fast scalar multiplication, in: International Conference on Cryptology in Africa, Springer, 2012, pp. 342–359.
 - [18] W. Yu, G. Xu, Pre-computation scheme of window τ naf for koblitz curves revisited, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2021, pp. 187–218.
 - [19] J. H. Cheon, W. Cho, J. Kim, D. Stehlé, Homomorphic multiple precision multiplication for ckks and reduced modulus consumption, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023, pp. 696–710.
 - [20] S. Narisada, H. Okada, K. Fukushima, S. Kiyomoto, T. Nishide, Gpu acceleration of high-precision homomorphic computation utilizing redundant representation, in: Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, 2023, pp. 1–9.
 - [21] E. Brier, M. Joye, Weierstrass elliptic curves and side-channel attacks, in: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, 2002, pp. 335–345.
 - [22] Z. Yan, D. Li, Performance analysis for resource constrained decentralized federated learning over wireless networks, IEEE Transactions on Communications (2024).

- [23] E. Illi, M. Qaraqe, S. Althunibat, A. Alhasanat, M. Alsafasfeh, M. de Ree, G. Mantas, J. Rodriguez, W. Aman, S. Al-Kuwari, Physical layer security for authentication, confidentiality, and malicious node detection: a paradigm shift in securing iot networks, *IEEE Communications Surveys & Tutorials* (2023).
- [24] X. Fu, Q. Li, W. Li, Modeling and analysis of industrial iot reliability to cascade failures: An information-service coupling perspective, *Reliability Engineering & System Safety* 239 (2023) 109517.
- [25] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, M. Guizani, A survey of machine and deep learning methods for internet of things (iot) security, *IEEE communications surveys & tutorials* 22 (3) (2020) 1646–1685.
- [26] X. Fan, X. Deng, Y. Xia, L. Yi, L. T. Yang, C. Zhu, Tensor-based confident information coverage reliability of hybrid internet of things, *IEEE Transactions on Mobile Computing* 23 (3) (2023) 2171–2185.
- [27] C. Patel, A. K. Bashir, A. A. AlZubi, R. Jhaveri, Ebake-se: A novel ecc-based authenticated key exchange between industrial iot devices using secure element, *Digital Communications and Networks* 9 (2) (2023) 358–366.
- [28] Y. Cao, A. Chandrasekar, T. Radhika, V. Vijayakumar, Input-to-state stability of stochastic markovian jump genetic regulatory networks, *Mathematics and Computers in Simulation* 222 (2024) 174–187.
- [29] M. Zeghid, H. Y. Ahmed, A. Chehri, A. Sghaier, Speed/area-efficient ecc processor implementation over gf (2 m) on fpga via novel algorithm-architecture co-design, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31 (8) (2023) 1192–1203.
- [30] J. Ye, Z. Yang, An ecc with error detection and against side channel attacks for resource constrained devices, *Journal of King Saud University-Computer and Information Sciences* 36 (4) (2024) 102019.
- [31] A. Sahasrabuddhe, D. S. Laiphrakpam, Multiple images encryption based on 3d scrambling and hyper-chaotic system, *Information Sciences* 550 (2021) 252–267.

- [32] J. Gao, M. Liu, P. Li, A. A. Laghari, A. R. Javed, N. Victor, T. R. Gadekallu, Deep incomplete multi-view clustering via information bottleneck for pattern mining of data in extreme-environment iot, *IEEE Internet of Things Journal* (2023).
- [33] S. Barbarossa, D. Comminiello, E. Grassucci, F. Pezone, S. Sardellitti, P. Di Lorenzo, Semantic communications based on adaptive generative models and information bottleneck, *IEEE Communications Magazine* 61 (11) (2023) 36–41.
- [34] R. Huo, X. Cheng, C. Sun, T. Huang, A cluster-based data transmission strategy for blockchain network in the industrial internet of things, *IEEE Transactions on Network and Service Management* (2024).
- [35] J. Zhang, Z. Chen, M. Ma, R. Jiang, H. Li, W. Wang, High-performance ecc scalar multiplication architecture based on comb method and low-latency window recoding algorithm, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 32 (2) (2023) 382–395.
- [36] A. M. Awaludin, H. T. Larasati, H. Kim, High-speed and unified ecc processor for generic weierstrass curves over $gf(p)$ on fpga, *Sensors* 21 (4) (2021) 1451.
- [37] D. F. Aranha, Y. El Housni, A. Guillevic, A survey of elliptic curves for proof systems, *Designs, Codes and Cryptography* 91 (11) (2023) 3333–3378.
- [38] A. Golovnev, S. Guo, S. Peters, N. Stephens-Davidowitz, Revisiting time-space tradeoffs for function inversion, in: *Annual International Cryptology Conference*, Springer, 2023, pp. 453–481.
- [39] Y. Sun, S. S. Chow, C. Chevalier, J. Wang, Elastic msm: A fast, elastic and modular preprocessing technique for multi-scalar multiplication algorithm on gpus, in: *Proceedings of the 31st USENIX Security Symposium (USENIX Security '22)*, USENIX Association, 2022, pp. 153–170.
- [40] P. C. Kocher, Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems, in: *Annual International Cryptology Conference*, Springer, 1996, pp. 104–113.

- [41] D. Boneh, R. A. DeMillo, R. J. Lipton, On the importance of checking cryptographic protocols for faults, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1997, pp. 37–51.
- [42] Q. Xiong, W. Ma, X. Shi, Y. Zhou, H. Jin, K. Huang, H. Wang, Z. Wang, gecc: A gpu-based high-throughput framework for elliptic curve cryptography, arXiv preprint arXiv:2501.03245 (2024).
- [43] H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, T. Stouraitis, A high-speed fpga implementation of an rsd-based ecc processor, *IEEE Transactions on very large scale integration (vlsi) systems* 24 (1) (2015) 151–164.
- [44] Y. Jiang, J. Zhang, A. Wang, Y. Hao, J. Wang, Z. Chen, L. Zhu, Low-latency and area-efficient elliptic curve point multiplication architectures over koblitz curves, *IEEE Internet of Things Journal* (2025).