A Lightweight Enhanced Montgomery Elliptic Curve Cryptography Scheme for Secure Communication in Wireless Sensor Networks

M. Nanak Zakaria
Electrical Engineering Department, Politeknik Negeri Malang, Indonesia | Electrical and Informatics Engineering Study Program, Universitas Negeri Malang, Indonesia
m.nanak.2205349@students.um.ac.id

Muladi
Electrical and Informatics Engineering Study Program, Universitas Negeri Malang, Indonesia
muladi@um.ac.id (corresponding author)

Hakkun Elmunsyah
Electrical and Informatics Engineering Study Program, Universitas Negeri Malang, Indonesia
Hakkun@um.ac.id

Received: 25 January 2026 | Revised: 14 March 2026, 20 March 2026, and 6 April 2026 | Accepted: 10 April 2026

 DOI: https://doi.org/10.48084/etasr.17754

ABSTRACT

This paper presents an Enhanced Montgomery Elliptic Curve Cryptography (ECC) scheme suitable for secure communication in resource-constrained Wireless Sensor Networks (WSNs). While Montgomery-based ECC is attractive for fast scalar multiplication, practical deployments on sensor nodes still suffer from significant encryption and decryption latency, non-negligible energy consumption, and limited CPU and memory capability. To address these issues, the proposed design revisits the scalar multiplication ladder and unifies the differential addition formulas to reduce redundant field operations, while preserving the original security level. The scheme's performance was compared with those of the classical Weierstrass-based ECC and the standard Montgomery implementation. Simulation results show that the proposed scheme shows a significant performance improvement in terms of execution time, energy consumption, and CPU encryption time, without increasing ciphertext size or memory requirements, while maintaining the level of cryptographic security and randomness. These findings indicate that the proposed scheme offers a practical public-key solution for long-lived, battery-powered WSNs.

Keywords-elliptic curve cryptography; Montgomery; CPU usage

I. INTRODUCTION

Elliptic Curve Cryptography (ECC) is one of the main building blocks of modern public-key security, especially in environments where bandwidth and processing power are scarce [1-3]. Montgomery curves are attractive because their arithmetic is efficient and well-suited to fast scalar multiplication, which underpins key exchange and digital signature operations [4]. In parallel, Wireless Sensor Networks (WSNs) are widely used in safety- and mission-critical settings, including environmental monitoring, industrial automation, and smart-city infrastructures, where secure yet energy-conscious communication is required despite the tight hardware limitations of the sensor nodes [5]. Despite this apparent match, deploying Montgomery-based ECC in WSNs remains non-trivial [6, 7]. Typical implementations still incur substantial encryption and decryption delays, high energy consumption, and notable memory and CPU overhead [8]. These drawbacks reduce node lifetime, threaten overall network robustness, and make applying strong public-key primitives in dense or long-lived sensor deployments difficult [9]. Furthermore, many existing optimization approaches concentrate on a single performance aspect, such as runtime or memory, rather than jointly optimizing time, energy, and resource usage under realistic WSN operating conditions [10].

The ciphertext overhead follows a nearly constant trend for all examined plaintext lengths [11]. This behavior comes from the ECIES design, where every ciphertext always includes the same helper data: an ephemeral public key, a nonce, and an authentication tag, regardless of the payload size [12]. Consequently, the overhead is effectively decoupled from the plaintext size and is mainly determined by the chosen curve format. When ECC is instantiated over a short Weierstrass curve (secp192r1), both coordinates of the public point must be transmitted, producing a larger and more inflexible overhead

[13]. In contrast, Montgomery-based variants only send a single u-coordinate, which substantially reduces the size of the encapsulation. The persistence of this overhead pattern across different plaintext sizes indicates that the ciphertext expansion is not driven by the payload length but by the structure of the key encapsulation primitive itself [14]. This property is especially relevant in constrained environments such as those found in WSNs, where the cost of radio transmission is a primary contributor to the overall energy use [15].

This article proposes the Enhanced Unified Differential Addition formulation of the Montgomery ladder for elliptic curve cryptography, reducing computational complexity by removing precomputation steps and intermediate variables, and consequently lowering the number of field arithmetic operations required per scalar-multiplication iteration. This simplification is achieved without modifying the ladder invariant and while maintaining the constant-time execution property, thus being consistent with the Montgomery ECC security principle. The resulting Enhanced-Montgomery implementation shows significant performance improvements in terms of execution time, energy consumption, and CPU time, without increasing ciphertext size or memory requirements, while maintaining the level of cryptographic security and randomness.

Encryption time grows only slightly as the plaintext becomes larger, and is mainly governed by the cost of the key-agreement phase rather than by Advanced Encryption Standard-Galois/Counter Mode (AES-GCM)'s per-byte processing [16]. On secp192r1, scalar multiplication is relatively heavy because it repeatedly performs point-doubling and point-addition over a prime field, which accounts for the noticeably longer encryption times of ECC-ECIES compared to Montgomery-based schemes [17]. The Montgomery ladder, with its streamlined and constant-time structure, supports much faster scalar multiplications, while the proposed Enhanced-Montgomery design goes further by eliminating arithmetic redundancy through symmetric update relations (t^+ and t^-), yielding an additional speedup. The almost flat time curves show that, after a certain point, encrypting more bytes adds very little to the total runtime, as AES processes data in a streaming fashion with minimal per-byte cost. Hence, the observed encryption times mainly capture the intrinsic efficiency of the different scalar-multiplication algorithms [18]. The energy-consumption trend closely tracks the encryption-time curve, since energy is obtained by multiplying execution time [19] with the (assumed) constant CPU power ($E = P \times t$). With the processor's power draw fixed during cryptographic routines, any change in runtime translates directly into a proportional change in energy [20]. As a result, ECC shows the largest energy usage because its scalar-multiplication phase is longer, whereas the standard and unified Montgomery schemes incur noticeably lower energy costs. The Enhanced-Montgomery ladder consistently achieves the lowest energy values, confirming that its arithmetic refinements reduce the amount of work per encryption. The strong similarity between the time and energy plots is thus expected and reflects [21] how the computational load is distributed under a fixed-power model typical of low-power embedded platforms and WSN nodes [22].

Peak memory requirements differ only slightly among the three schemes because, in all cases, the ECIES processing pipeline reserves a comparable stack footprint during execution [23]. The measured memory mainly reflects temporary buffers, scalar-multiplication intermediates, ephemeral key material, and the AES working space [24]. Although ECC and Montgomery arithmetic have different intrinsic complexities, the Python implementation relies on high-level numeric objects whose allocations are broadly similar, which hides the finer memory distinctions that would be visible in a C or firmware-level realization [25]. The only consistent difference is a modest reduction for Enhanced-Montgomery, stemming from fewer intermediate variables and a more compact algebraic flow. Overall, the nearly flat memory curve suggests that high-level language overhead can dominate and obscure algorithmic memory optimizations, an important caveat when interpreting simulation results versus actual embedded deployments [26]. CPU time measures how long the processor is actively executing the encryption algorithm, ignoring any idle periods or delays from the operating system [27]. It therefore captures the intrinsic computational effort of the scheme. In line with expectations, ECC exhibits the highest CPU time because scalar multiplication on short Weierstrass curves is relatively heavy, involving many field operations and conditional branches [28]. Montgomery-based designs, by contrast, benefit from the ladder's regular and lightweight arithmetic pattern, leading to much lower CPU times. The Enhanced-Montgomery variant achieves the smallest CPU-time values, indicating that its algebraic streamlining cuts down the number of multiplications and subtractions per ladder step. These CPU-time results align well with the theoretical operation counts of each method and serve as a good proxy for their behavior on constrained microcontrollers, where CPU cycles are tightly tied to energy usage and device lifetime [29].

II. THE PROPOSED SCHEME

The proposed solution introduces a mathematically streamlined unified differential-addition step inside the Montgomery ladder. Instead of the classical scheme that relies on several intermediate variables (A, B, C, D, DA, CB) and multiple modular multiplications per iteration, the Enhanced-Montgomery design directly updates the projective x-coordinates as:

$$X_3=(X_1Z_2+Z_1X_2)^2,\ Z_3=(X_1Z_2-Z_1X_2)^2\cdot u \quad (1)$$

where ($X_1 : Z_1$) ($X_1 : Z_1$) encode the difference point P and $u = x(P)$.

These formulas are algebraically equivalent to the standard differential-addition rules under projective equivalence, so they produce the same point on the Montgomery curve while eliminating the precomputation block and several field multiplications. Consequently, each ladder step requires fewer finite-field operations, reducing scalar-multiplication time and energy consumption in wireless sensor nodes, without changing the curve parameters, key sizes, or security level of the conventional ECC and standard Montgomery implementations. The block diagram of the Enhanced Montgomery ECC, which is an improvement of the Montgomery ECC and the Classic ECC, is shown in Figure 1.

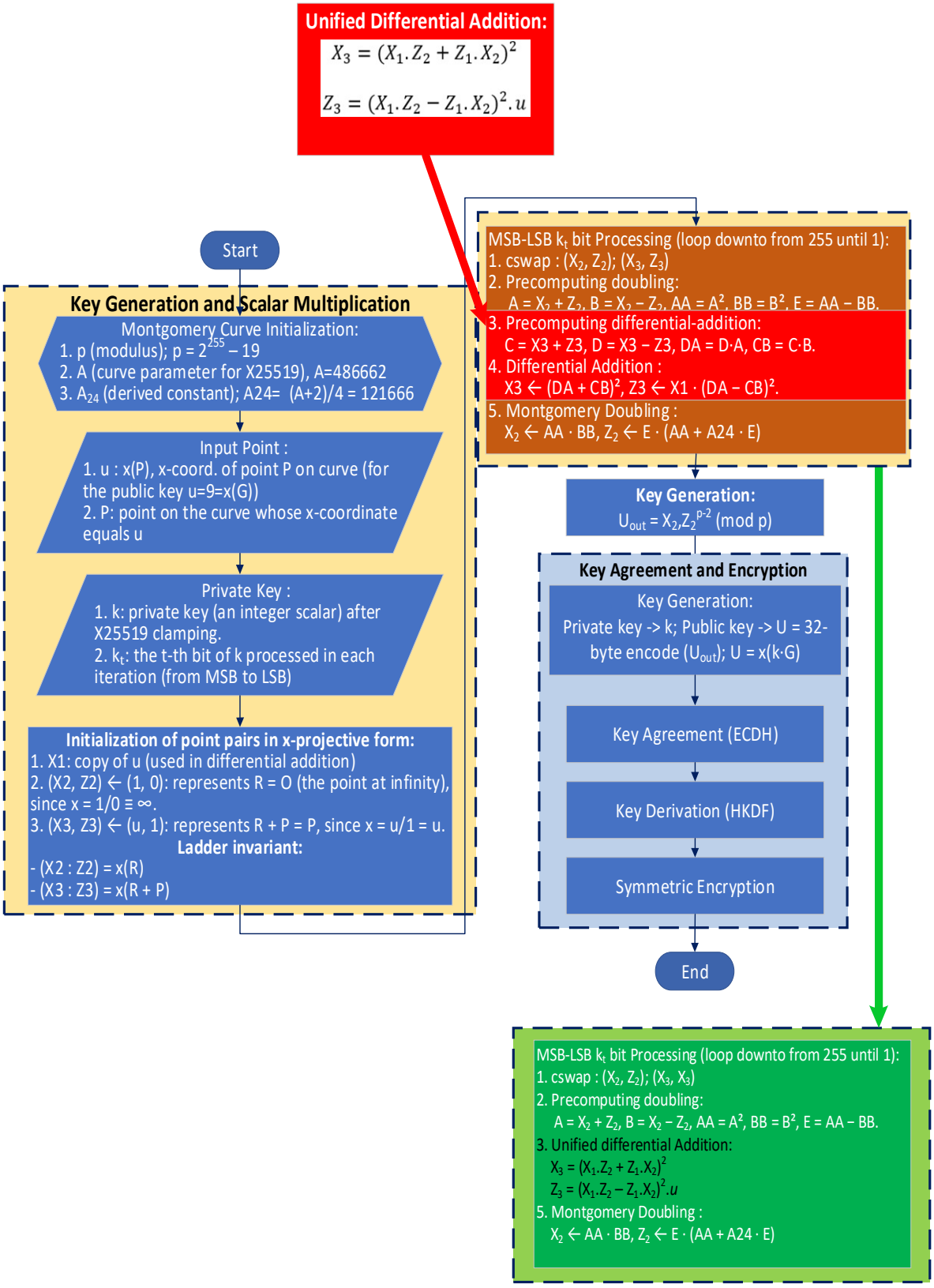

Fig. 1. Blok diagram of the Enhanced Montgomery ECC.

Figure 1 shows the Montgomery ladder-based Montgomery ECC workflow for scalar multiplication, which is executed by processing bits from the MSB to the LSB. In conventional implementations, each ladder iteration involves a precomputation step for differential addition (Step 3) and a differential addition update (Step 4), followed by a doubling operation in the next step. The contribution of this research is to replace the conventional calculation sequence in Steps 3 and 4 with a single "Unified Differential Addition" block, thereby eliminating the precomputation step and intermediate variables. As a result of this replacement, the main loop block of the ladder is represented in a more compact form, as indicated by the green box in the Figure, without changing the bit processing mechanism, ladder invariants, or constant-time execution properties.

A. Optimization of the Unified Differential Addition Formula

In the conventional Montgomery ladder implementation [7], the differential addition process requires a number of intermediate variables to compute the result of point addition in projective coordinates. The formula steps for the intermediate variable computation stage have a form that has been proven secure against side-channel attacks because it is executed in constant time. However, this method still requires many modular operations consisting of eight modular multiplications and several addition and subtraction operations, which consume significant computational and memory resources. We propose an optimized form of the unified differential addition formula as follows:

$$X_3 = (X_1.Z_2 + Z_1.X_2)^2 \quad (2)$$

$$Z_3 = (X_1.Z_2 - Z_1.X_2)^2.u \quad (3)$$

where u = x(P) represents the x-coordinate of the difference point $P = R_1 - R_2$.

This formula directly combines addition and subtraction operations without requiring intermediate variables such as A, B, C, D, DA, and CB. Thus, the entire differential addition precomputation block can be completely eliminated.

B. Arithmetic Simplification of the Differential Addition

This study simplifies the differential addition stage of the Montgomery ladder by rearranging the conventional algebraic expressions into a reduced form. This rearrangement eliminates intermediate variables and precomputation steps without changing the ladder structure, computation invariants, or constant-time execution properties, thus remaining consistent with the Montgomery ECC security principles. As a starting point, scalar multiplication in the Montgomery ladder is represented in projective coordinates (X: Z), with the affine mapping x = X/Z. At each ladder iteration, two points are simultaneously maintained, *R* and R+P, with the principal invariant:

$$R_1 - R_0 = P \quad (4)$$

This invariant constrains the space of allowable transformations, so any algebraic reformulation must remain consistent with this relationship.

In the conventional Montgomery formulation, the differential addition process is realized through a number of intermediate variables and pre-computation steps, which directly increase the burden of field arithmetic operations at each iteration. The analysis in this study shows that this complexity is not an inherent consequence of the geometry of the Montgomery curve, but rather stems from the algebraic representation used in the calculation. Based on this observation, simplification is directed at restructuring the differential addition expression so that the calculation result can be obtained directly from the coordinate pairs (X_2, Z_2) and (X_3, Z_3), without involving intermediate variables or additional pre-computation steps. In every iteration, the ladder maintains an invariant:

$$x(R) = (X_2 : Z_2) \; ; \; x(R+P) = (X_3 : Z_3) \quad (5)$$

with the difference between the two being P, where:

$$x(P) = (X_1 : Z_1) \Rightarrow u = X_1/Z_1 \quad (6)$$

In projective coordinates, two pairs (X:Z) that differ only by a non-zero scale factor λ represent the same point:

$$(X:Z) \sim (\lambda X : \lambda Z)$$

The resulting symmetric expression directly combines the addition and subtraction operations of coordinates in a finite field. By rearranging the arithmetic formulation at the differential addition stage, two main nodes are obtained that represent the calculation results without involving intermediate variables:

$$X_3(new) = (X_1Z_2 + Z_1X_2)^2$$

$$Z_3(new) = (X_1Z_2 - Z_1X_2)^2 . U \quad (7)$$

where $u = X_1/Z_1 = x(P)$

These equations represent the results of the unified differential addition, where addition and subtraction operations are combined directly without the need for precomputation or intermediate variables. Because this formulation maintains ladder invariants and does not introduce conditional branching, the constant-time execution property of the Montgomery ladder is also maintained:

$$X_3Z_2 - Z_3X_2 \approx 1/Z1 \; (X_1Z_2 - Z_1X_2) \quad (8)$$

$$Z_3(new) = X_1.(1/Z_1)^2(X_1Z_2 - Z_1X_2)^2 \Rightarrow$$

$$(1/Z_1)^2 \approx (1/Z_1)$$

$$= (X_1/Z_1).(X_1Z_2 - Z_1X_2)^2 = (X_1Z_2 - Z_1X_2)^2 . u$$

where $u = X_1/Z_1 = x(P)$ and therefore:

$$Z_3(new) = (X_1Z_2 - Z_1X_2)^2 . U \quad (9)$$

III. RESULTS

This section compares three elliptic-curve configurations for WSNs: a baseline Weierstrass ECC, a standard Montgomery scheme, and the proposed Enhanced-Montgomery scheme. The analysis focuses on WSN-relevant metrics, i.e. encryption and decryption latency, energy consumption, peak memory, and CPU time, with all

variants implemented on the same software stack and sensor platform, and each value averaged over multiple runs.

Before running the simulation, several general configuration parameters were defined to ensure a consistent and reproducible experimental environment. The processor power consumption was assumed to be constant at 0.015 W, representing a typical power level for low-power embedded processors commonly used in wireless sensor nodes. To obtain statistically stable measurements, each experimental configuration was repeated 30 times, and the reported values correspond to the averages of these runs. The plaintext payload size used in the experiments was varied to evaluate the scalability of the cryptographic algorithms with respect to message length. Specifically, the payload size ranged from 16 × 10240 bytes (approximately 160 KB) to 16 × 10240 × 24 bytes (approximately 3.84 MB). This range allows the evaluation of encryption time, decryption time, energy consumption, memory usage, and CPU time across different communication loads that may occur in WSN environments. For the classical elliptic-curve baseline, the experiments used the secp192r1 curve (NIST P-192), which is widely adopted in cryptographic applications. This elliptic curve is defined over a prime finite field, where the parameter p denotes the prime modulus that determines the arithmetic field used in the elliptic-curve computations. The parameters a and b represent the coefficients of the elliptic-curve equation that define the curve structure. The base point G, defined by the coordinates (Gx, Gy), represents the generator point of the elliptic-curve group. This point is used as the starting element for scalar multiplication operations during key generation and key agreement. The parameter n denotes the order of the base point, indicating the number of points in the cyclic subgroup generated by G. The curve is therefore specified by the following domain parameters:

- p = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFFFFFFFFFFFF
- a = −3(modp)
- b = 64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1
- Gx = 188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012
- Gy = 07192B95FFC8DA78631011ED6B24CDD573F977A11E794811
- n = FFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831

These parameters follow the official NIST P-192 domain parameter specification and serve as the reference configuration for the baseline ECC implementation used in the comparative evaluation.

The results indicate that ciphertext overhead differs markedly between the classical ECC and the two Montgomery-based schemes, while standard and Enhanced Montgomery exhibit nearly identical overhead, as illustrated in Figure 2. The large ciphertext overhead of the classic ECC at 76 bytes is still much larger than the standard Montgomery ECC's, which is the same with that of Enhanced Montgomery at 60 bytes.

Regarding encryption time at the sender, the Enhanced Montgomery ECC is still the fastest as shown in Figure 3.

The difference in time required for encryption is shown in Figure 4. The implementation of the enhanced Montgomery shows that the encryption time is 7.038 ms faster than that of the standard Montgomery and 259.765 ms faster than the one of ECC.

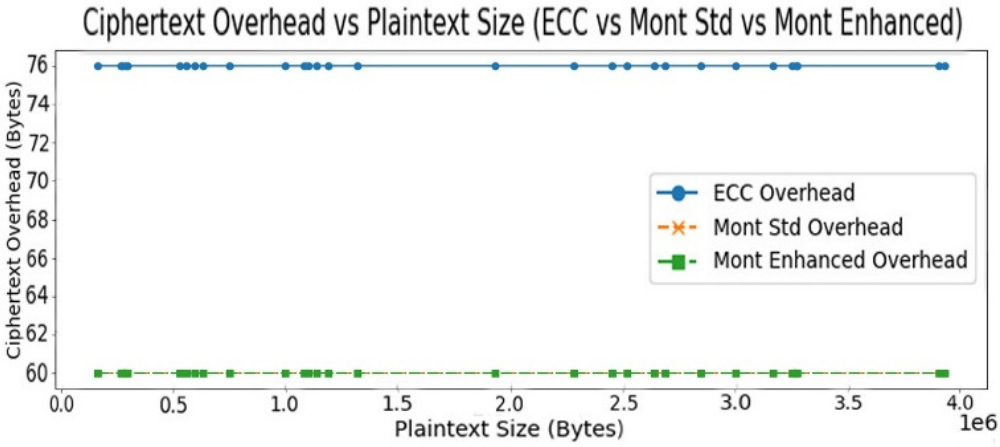

Fig. 2. Ciphertext overhead.

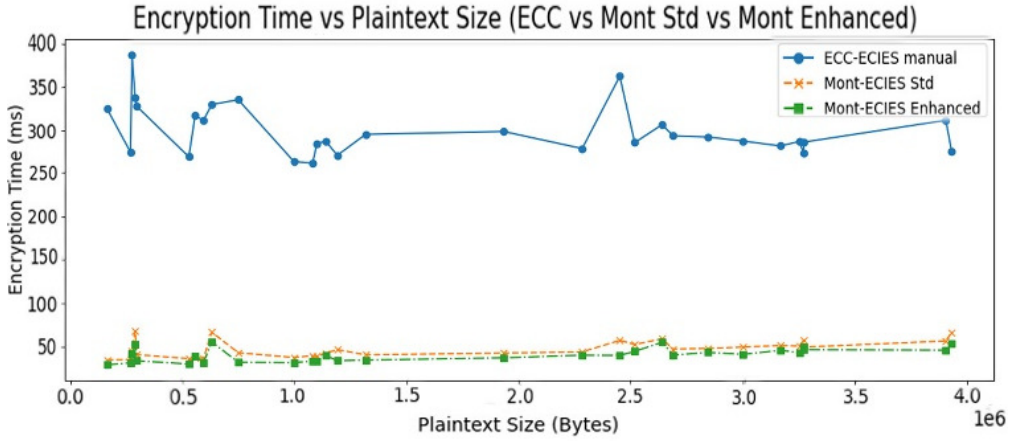

Fig. 3. Encryption time.

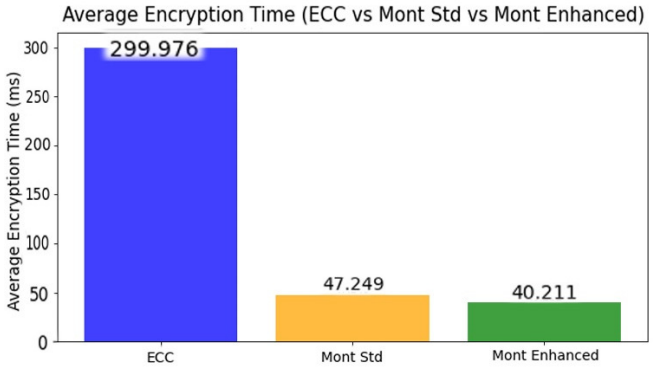

Fig. 4. Average encryption time.

Over the entire range of payloads, ECC-ECIES exhibits the highest energy demand, with values clustering around 4.0×10^{-3}–5.0×10^{-3} J per encryption, and only modest variation as the plaintext grows. In contrast, both Montgomery-based variants operate in a much lower energy regime, roughly an order of magnitude below the Weierstrass implementation. The standard Montgomery scheme consistently outperforms ECC, but the Enhanced-Montgomery design further reduces the energy cost for almost all data sizes, reflecting the benefits of its optimized scalar-multiplication routine, as shown in Figure 5.

The bar chart in Figure 6 summarizing the averages makes this contrast explicit: the baseline ECC configuration requires approximately 4.5×10^{-3} J per encryption, whereas the standard Montgomery scheme lowers this figure to about 7.09×10^{-4} J, and the Enhanced-Montgomery variant achieves the smallest average energy consumption of roughly 6.03×10^{-4} J. This corresponds to an energy saving of around 84–87% compared to the Weierstrass-based ECC baseline, which is highly significant for battery-powered sensor nodes.

Figure 8 shows that peak memory usage is essentially the same for all three schemes and is driven almost entirely by the plaintext size rather than by the curve choice.

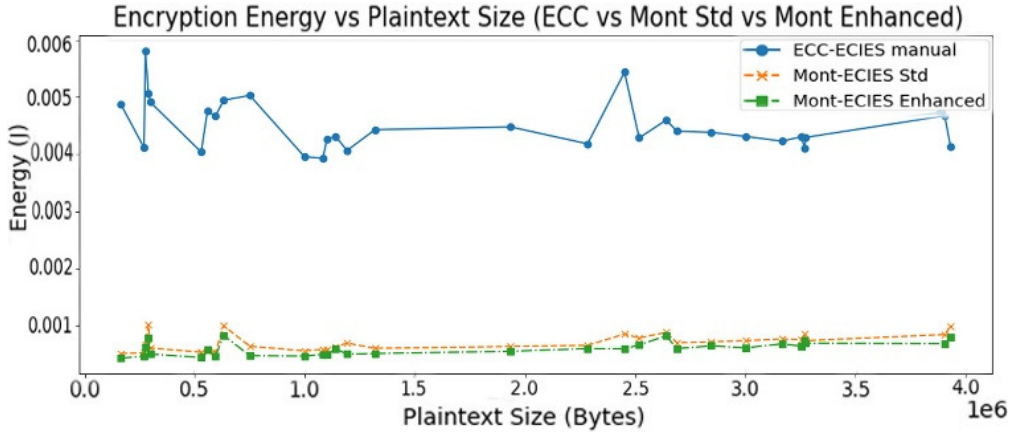

Fig. 5. Encryption energy consumption.

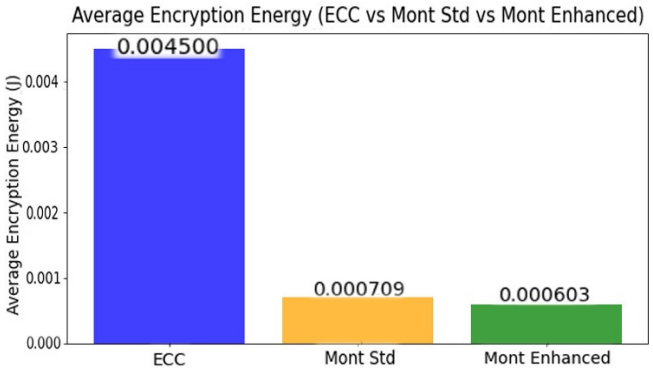

Fig. 6. Average encryption energy.

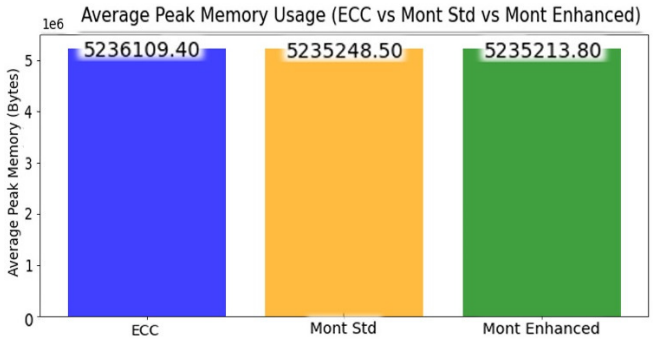

Fig. 7. Average peak memory usage.

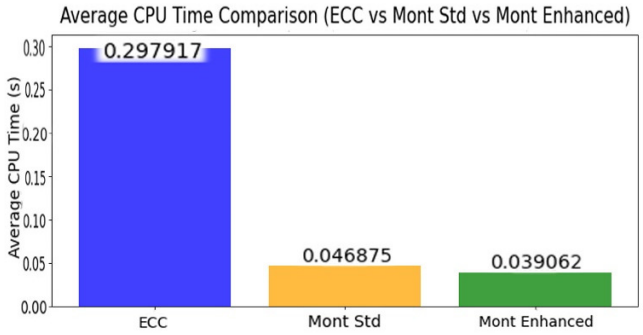

Fig. 8. Average CPU time comparison.

The bar in Figure 7 summarizes average peak memory: The acquired values of standard ECC, Montgomery, and Enhanced-Montgomery all lie around 5.23×10^6 bytes, with only marginal differences. The line plot reinforces this observation: for each plaintext length, the three curves are practically indistinguishable and increase linearly with payload size. This indicates that the ECIES implementation allocates a similar working set of buffers, intermediate values, and encryption workspace regardless of the underlying elliptic-curve formulation. Any algorithmic savings in temporary storage for Enhanced-Montgomery are therefore overshadowed by the dominant, payload-proportional buffer requirements of the high-level implementation.

The CPU-time measurements provide a clear view of the computational effort required by each elliptic-curve configuration during encryption. Across all tested plaintext sizes, the standard Weierstrass-based ECC implementation consistently exhibits the highest CPU times, reflecting the relatively heavy scalar multiplication on short-Weierstrass curves. In contrast, both Montgomery-based schemes operate in a substantially lower CPU-time regime, due to the regular ladder structure that minimizes control-flow complexity and arithmetic overhead (Figure 9). Among them, the Enhanced-Montgomery variant achieves the smallest CPU-time values for nearly all data lengths, indicating that its algebraic refinements effectively reduce the number of field operations per scalar-multiplication step.

In Figure 10, the near-flat profile of all three curves as the payload grows suggests that CPU time is dominated by the key-agreement phase rather than by the symmetric-data path, so changes in plaintext size have only a marginal impact on the total processor workload. These results confirm that the observed CPU-time differences mainly capture the intrinsic efficiency of the underlying scalar-multiplication algorithms, a factor that is directly relevant for WSN nodes where available CPU cycles are tightly linked to energy budget and device lifetime.

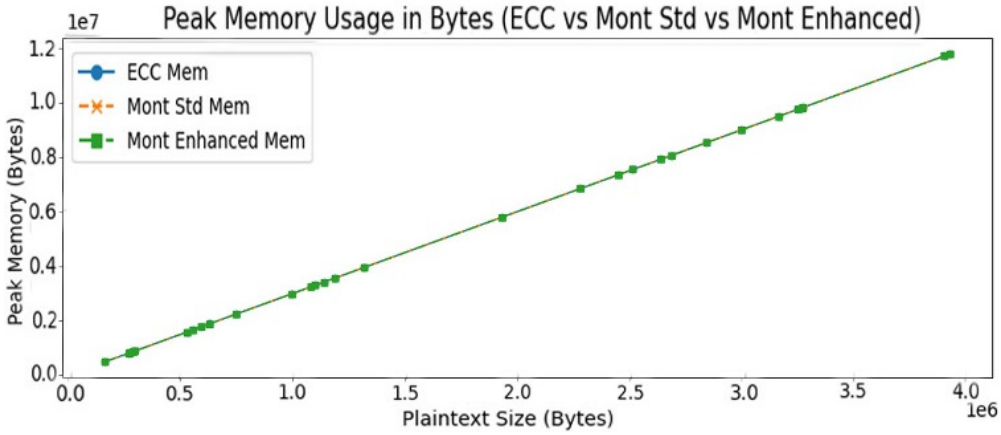

Fig. 9. Peak memory usage.

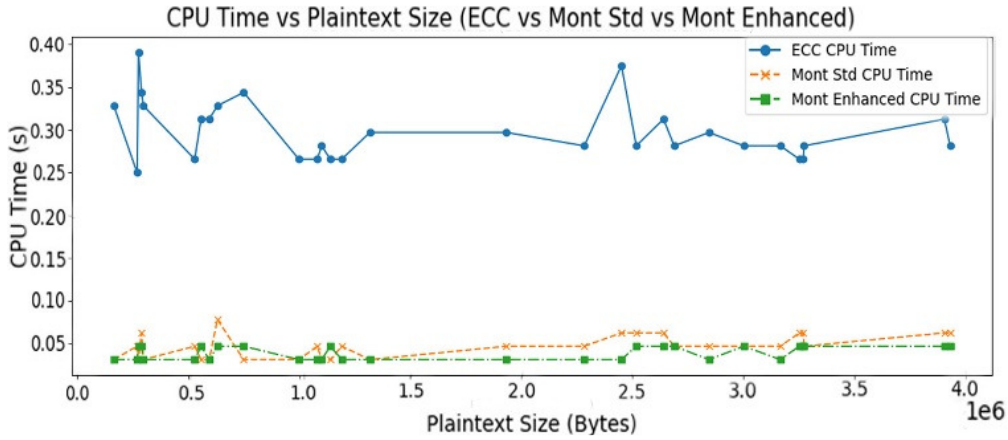

Fig. 10. CPU time.

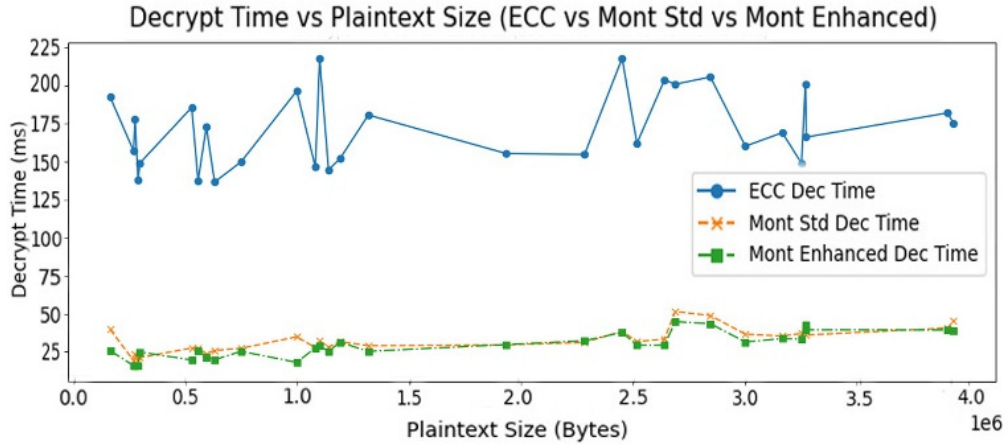

Fig. 11. Decryption time comparison.

Figure 11 confirms this trend for individual plaintext sizes. ECC decryption times fluctuate between roughly 135 ms and 220 ms, with no strong dependence on payload length. By contrast, the Montgomery-based curves lie in a much lower band: the standard Montgomery scheme generally ranges from 20 to 45 ms, and the Enhanced-Montgomery implementation from 18 to 42 ms, with the enhanced curve almost always lying slightly below the standard one. The relatively flat shapes of all three curves indicate that decryption cost is dominated by the scalar-multiplication and key-agreement phase rather than by the symmetric data processing, so increasing the plaintext size has only a modest effect on total runtime.

The bar chart in Figure 12 shows that standard Weierstrass-based ECC has by far the slowest average decryption time, at about 171.381 ms per message. In comparison, the conventional Montgomery scheme reduces the mean decryption time to roughly 32.561 ms, while the Enhanced-Montgomery design further lowers it to around 29.080 ms. Thus, both Montgomery variants are more than five times faster than the baseline ECC implementation, and the enhanced version offers the best overall performance.

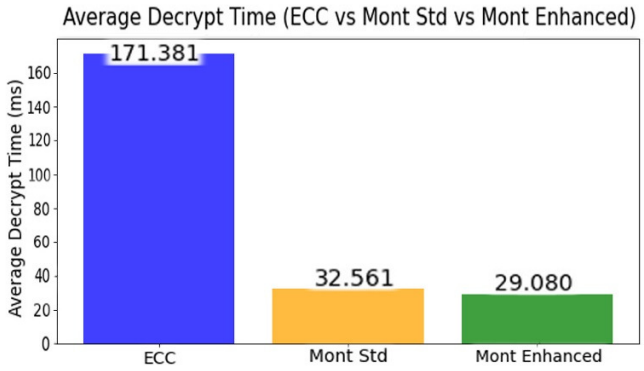

Fig. 12. Average Decryption time.

Taken together, these results show that the Enhanced-Montgomery approach is the most efficient option in terms of decryption time, followed closely by the standard Montgomery scheme, while traditional ECC is clearly the least suitable for latency-sensitive and resource-constrained deployments such as WSNs.

Figure 13 confirms this pattern for individual payload lengths. The ECC curve fluctuates in the range of approximately 0.0020–0.0033 J, with no strong dependence on

plaintext size. In contrast, the two Montgomery curves remain clustered between about 0.00025 J and 0.00075 J, with the Enhanced-Montgomery line generally lying slightly below the standard Montgomery line. The relatively flat profiles across growing plaintext sizes indicate that decryption energy is dominated by the elliptic-curve key-agreement cost rather than by the symmetric data path, so increasing the payload contributes only marginally to the total energy budget.

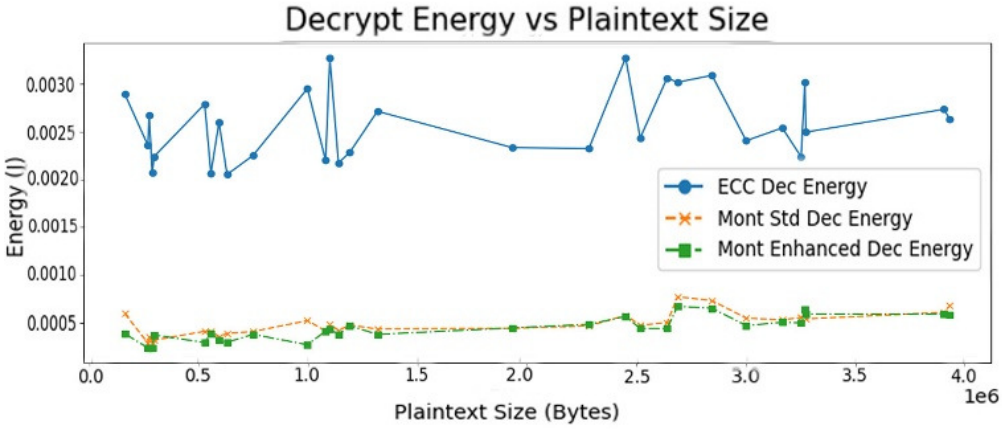

Fig. 13. Decryption energy.

Standard ECC shows the highest average decryption energy at about 0.002571 J per message (Figure 14). The Montgomery implementation lowers this to roughly 0.000488 J, while the Enhanced-Montgomery scheme achieves the smallest average value of around 0.000436 J. Thus, both Montgomery variants reduce decryption energy by almost an order of magnitude compared with the Weierstrass-based ECC baseline, and the enhanced version is the most economical overall.

Overall, these results show that the Enhanced-Montgomery offers the best decryption-energy efficiency, followed closely by standard Montgomery, while classical ECC is clearly the least suitable choice for energy-constrained deployments such as WSNs.

The line graph in Figure 15 confirms this: the three curves overlap from roughly 4×10^5 up to about 1.18×10^7 bytes as the payload increases, indicating that memory is dominated by ECIES buffers and the runtime environment rather than by the specific elliptic-curve arithmetic. Consequently, all three approaches can be regarded as effectively equivalent with respect to peak memory requirements in WSN-like settings.

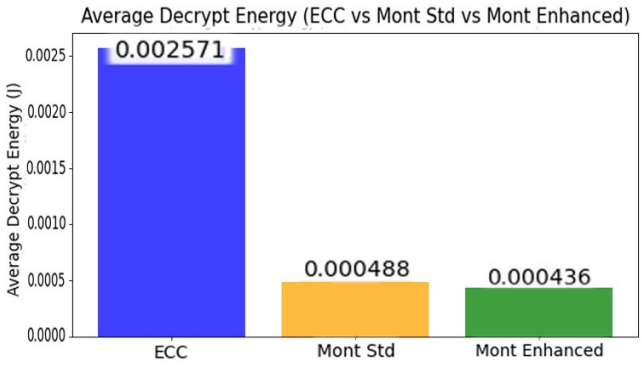

Fig. 14. Average decryption energy.

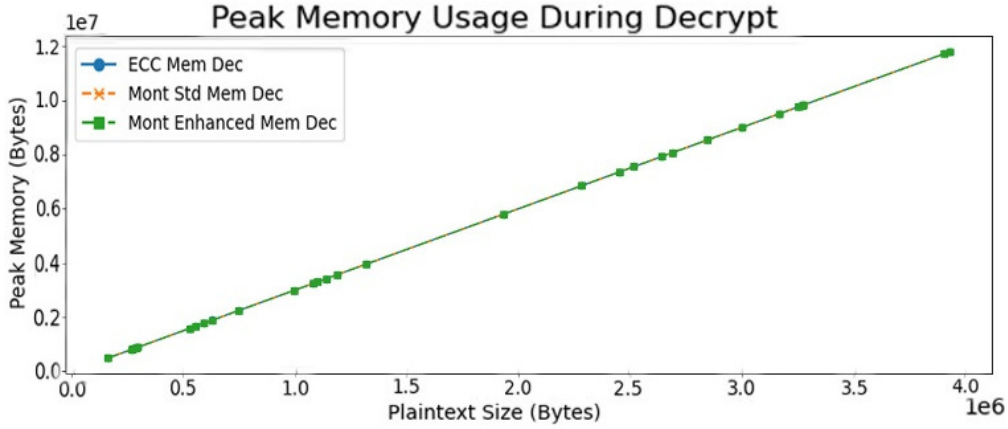

Fig. 15. Peak decryption memory usage.

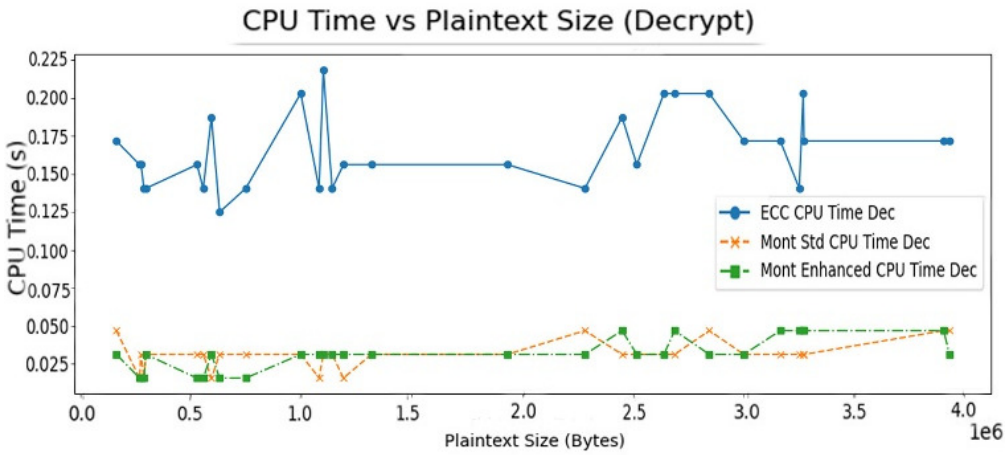

Fig. 16. CPU time vs plaintext size (decryption).

From the bar chart in Figure 17, the average peak memory is about 5235791.5 bytes for standard ECC, 5235297.2 bytes for Montgomery, and 5235283.9 bytes for Enhanced-Montgomery. The differences are extremely small well below 1% so no method offers a meaningful advantage in terms of peak memory.

In Figure 16, the ECC curve is clearly the highest, with CPU times roughly in the range of 0.14–0.22 s across all payload sizes. In contrast, both Montgomery variants stay in a much lower band, around 0.02–0.05 s, and show only modest variation with plaintext length. This indicates that the bulk of the CPU cost comes from the elliptic-curve computations, not from processing additional bytes of data.

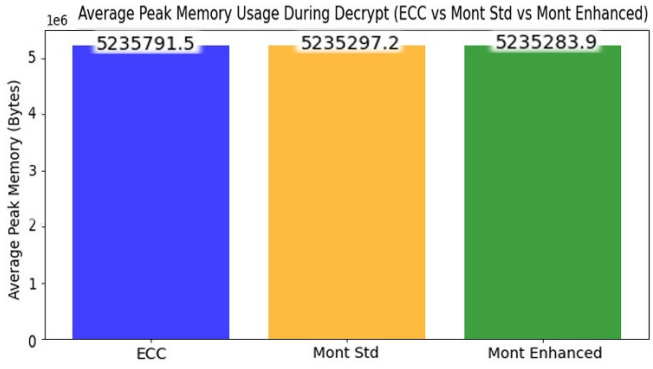

Fig. 17. Average decryption peak memory usage.

Figure 18 summarizes the averages: standard ECC requires about 0.166 s of CPU time per decryption, whereas both the standard Montgomery and Enhanced-Montgomery schemes need only about 0.031 s on average. This corresponds to roughly a reduction in CPU time compared with the Weierstrass-based ECC baseline. Since lower CPU time directly translates to fewer processor cycles and reduced energy use in WSN nodes, the Montgomery-based approaches are clearly superior, with Enhanced-Montgomery matching the best CPU-time performance.

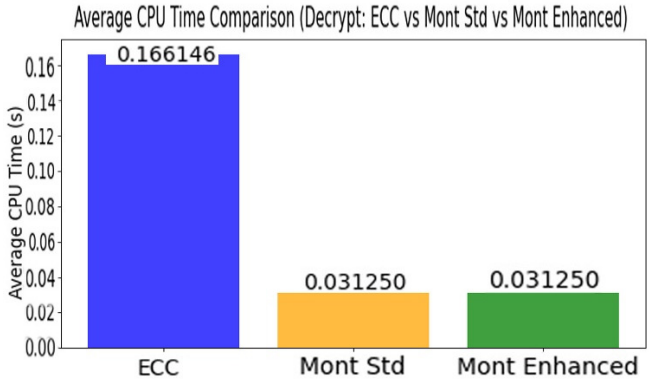

Fig. 18. Average CPU time.

The evaluation of the randomness of the encrypted bitstream was carried out using the NIST Statistical Test Suite (SP 800-22) at a significance level of α=0.01. The results can be seen Table I.

It can be seen that all three schemes pass 18 randomness tests at α = 0.01, with p-values well above the 0.01 threshold. This means that statistically, all three schemes have equivalent levels of randomness (entropy, absence of global patterns, and low linear complexity), so the Montgomery Enhanced modification only improves computational efficiency without reducing the quality of randomness or the level of cryptographic security compared to standard ECC and regular Montgomery.

IV. CONCLUSION

This study introduced an Enhanced-Montgomery elliptic-curve scheme that systematically improves the practicality of public-key cryptography in Wireless Sensor Networks (WSNs). By unifying and simplifying the differential-addition stage in the Montgomery ladder, the proposed design reduces the number of field operations while keeping the underlying security level unchanged.

TABLE I. NIST STATISTICAL TEST SUITE (SP 800-22) RESULTS

No	test	p_value ECC Standard	p_value ECC Montgomery	p_value Enhanced Montgomery	pass
1	Frequency (monobit)	0.741199834	0.205588037	0.851366808	TRUE
2	Block frequency	0.651761059	0.163747486	0.434633235	TRUE
3	Cumulative sums (forward)	0.878035869	0.210035578	0.666983069	TRUE
4	Cumulative sums (backward)	0.582882221	0.375060955	0.83956174	TRUE
5	Runs	0.633272726	0.386656072	0.426434315	TRUE
6	Longest Run	0.348047384	0.941262041	0.744795251	TRUE
7	Rank (32×32)	0.075793112	0.051507947	0.072578924	TRUE
8	FFT (spectral)	0.674054877	0.424938399	0.638304582	TRUE
9	Non-overlapping template (m=9)	0.82408476	0.303807751	0.369094395	TRUE
10	Universal (Maurer)	0.614775409	0.656649092	0.939418814	TRUE
11	Approximate entropy (m=2)	0.717593795	0.178856283	0.1579412	TRUE
12	Random excursions	0.113927666	0.094812674	0.117903348	TRUE
13	Random excursions variant	0.054659597	0.179156939	0.072813333	TRUE
14	Serial (p1) m=2	0.845018377	0.307914434	0.716174854	TRUE
15	Serial (p2) m=2	0.633220396	0.385326014	0.426420561	TRUE
16	Linear Complexity (M=500)	0.581483279	0.511178699	0.259300032	TRUE

Across all experiments, Enhanced Montgomery consistently achieved the lowest encryption and decryption times, the smallest CPU time, and the lowest energy consumption, while maintaining ciphertext overhead and peak memory usage essentially identical to the values of standard Montgomery ECC and clearly below those of the Weierstrass-based baseline. These results show that the main cost driver in ECC-ECIES for WSNs is the scalar-multiplication routine, and that careful algebraic engineering of this step can yield performance gains of about one order of magnitude in time and energy costs without weakening randomness or statistical security. Consequently, the Enhanced Montgomery emerges as a compelling choice for long-lived, battery-powered WSN deployments that demand strong cryptographic protection under strict latency and resource constraints.

DECLARATION OF COMPETING INTERESTS

Not applicable to this work.

ACKNOWLEDGMENT

The authors would like to thank the research supervisors from the State University of Malang. Their collective support was critical in helping me overcome academic challenges.

DATA AVAILABILITY

No external dataset was used in this simulation study. The simulation parameters are mentioned within this paper.

REFERENCES

[1] A. Mishra, "Elliptic Curve Cryptography (ECC) for Security in wireless Sensor Network," *International journal of engineering research and technology*, May 2012.

[2] S. Urooj, S. Lata, S. Ahmad, S. Mehfuz, and S. Kalathil, "Cryptographic Data Security for Reliable Wireless Sensor Network," *Alexandria Engineering Journal*, vol. 72, pp. 37–50, June 2023, https://doi.org/10.1016/j.aej.2023.03.061.

[3] A. Aripriharta, M. Muladi, N. Mufti, I. A. E. Zaeni, I. M. Wirawan, and G.-J. Horng, "Modeling and Analysis of the Self-powered Device for Wireless Heart Rate Measurement." ENGINEERING, July 24, 2019, https://doi.org/10.20944/preprints201907.0272.v1.

[4] A. Sajid, O. S. Sonbul, M. Rashid, and M. Y. I. Zia, "A Hybrid Approach for Efficient and Secure Point Multiplication on Binary Edwards Curves," *Applied Sciences*, vol. 13, no. 9, May 2023, https://doi.org/10.3390/app13095799.

[5] R. Senthil Kumaran, K. Loga, M. Arul Pugazhendhi, and K. Saundariya, "Security Techniques in Wireless Sensor Networks - A Comprehensive Survey," in *2021 Smart Technologies, Communication and Robotics (STCR)*, July 2021, pp. 1–6, https://doi.org/10.1109/STCR51658.2021.9588872.

[6] S. Baktır and B. Sunar, "Finite Field Polynomial Multiplication in the Frequency Domain with Application to Elliptic Curve Cryptography," in *Computer and Information Sciences – ISCIS 2006*, 2006, pp. 991–1001, https://doi.org/10.1007/11902140_103.

[7] C. Costello and B. Smith, "Montgomery curves and their arithmetic," *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 227–240, Sept. 2018, https://doi.org/10.1007/s13389-017-0157-6.

[8] R. I. Lestari, V. Suryani, and A. A. Wardhana, "Digital Signature Method to Overcome Sniffing Attacks on LoRaWAN Network," *International journal of electrical and computer engineering systems*, vol. 13, no. 7, pp. 533–539, Sept. 2022, https://doi.org/10.32985/ijeces.13.7.5.

[9] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes, "CSIDH: An Efficient Post-Quantum Commutative Group Action," in *Advances in Cryptology – ASIACRYPT 2018*, 2018, pp. 395–427, https://doi.org/10.1007/978-3-030-03332-3_15.

[10] C. Anuradha *et al.*, "A Robust Security System Using SHA-512 with Reinforcement Learning in Wireless Sensor Networks," *Engineering, Technology & Applied Science Research*, vol. 15, no. 6, pp. 30080–30086, Dec. 2025, https://doi.org/10.48084/etasr.14048.

[11] E. T. Oladipupo *et al.*, "An Efficient Authenticated Elliptic Curve Cryptography Scheme for Multicore Wireless Sensor Networks," *IEEE Access*, vol. 11, pp. 1306–1323, 2023, https://doi.org/10.1109/ACCESS.2022.3233632.

[12] F. Rezaei and A. Zahedi, "Dealing with Wormhole Attacks in Wireless Sensor Networks Through Discovering Separate Routes Between Nodes," *Engineering, Technology & Applied Science Research*, vol. 7, no. 4, pp. 1771–1774, Aug. 2017, https://doi.org/10.48084/etasr.1118.

[13] K. Abhishek and E. G. D. P. Raj, "Computation of Trusted Short Weierstrass Elliptic Curves for Cryptography," *Cybernetics and Information Technologies*, vol. 21, no. 2, pp. 70–88, July 2021, https://doi.org/10.2478/cait-2021-0020.

[14] D. Cervantes-Vázquez, E. Ochoa-Jiménez, and F. Rodríguez-Henríquez, "Parallel Strategies for SIDH: Toward Computing SIDH Twice as Fast," *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1249–1260, June 2022, https://doi.org/10.1109/TC.2021.3080139.

[15] A. Aripriharta, A. Firmansah, M. Yazid, I. D. Wahyono, Muladi, and G. J. Horng, "Modelling of adaptive power management circuit with feedback for self-powered IoT," *Journal of Physics: Conference Series*, vol. 1595, no. 1, Apr. 2020, Art. no. 012023, https://doi.org/10.1088/1742-6596/1595/1/012023.

[16] A. Iqbal and T. Iqbal, "Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module," in *2018 IEEE Electrical Power and Energy Conference (EPEC)*, July 2018, pp. 1–5, https://doi.org/10.1109/EPEC.2018.8598380.

[17] A. Faz-Hernández, J. López, E. Ochoa-Jiménez, and F. Rodríguez-Henríquez, "A Faster Software Implementation of the Supersingular Isogeny Diffie-Hellman Key Exchange Protocol," *IEEE Transactions on Computers*, vol. 67, no. 11, pp. 1622–1636, Aug. 2018, https://doi.org/10.1109/TC.2017.2771535.

[18] S. Liu and L. Zhang, "Efficient Septuple Formula for Elliptic Curve and Efficient Scalar Multiplication Using a Triple-Base Chain Representation," *IEEE Access*, vol. 9, pp. 129512–129520, 2021, https://doi.org/10.1109/ACCESS.2021.3113792.

[19] S. D. Chavan and A. V. Kulkarni, "Event Based Clustering Localized Energy Efficient Ant Colony Optimization (EBC_LEE-ACO) for Performance Enhancement of Wireless Sensor Network," *Engineering, Technology & Applied Science Research*, vol. 8, no. 4, pp. 3177–3183, Aug. 2018, https://doi.org/10.48084/etasr.2121.

[20] N. Haidar Hari, M. Sholihul Hadi, and Sujito, "LEACH Protocol with Angular Area Routing: Boosting Energy Efficiency and QoS in Wireless Sensor Networks," Jan. 2024, https://doi.org/10.12785/ijcds/160188.

[21] N. H. Hari, Mokh. S. Hadi, S. Sujito, A. I. C. Ani, S. Setumin, and Mhd. Irvan, "ARZSEP: Angle-Based Routing Optimization in ZSEP Protocol for Heterogeneous WSNs," *Wireless Personal Communications*, vol. 139, no. 2, pp. 1013–1038, Nov. 2024, https://doi.org/10.1007/s11277-024-11651-w.

[22] Aripriharta, W. Z. Hao, Muladi, G.-J. Horng, and G.-J. Jong, "A New Bio-Inspired for Cooperative Data Transmission of IoT," *IEEE Access*, vol. 8, pp. 161884–161893, 2020, https://doi.org/10.1109/ACCESS.2020.3021507.

[23] T. Widiyaningtyas, I. Hidayah, and T. B. Adji, "Recommendation Algorithm Using Clustering-Based UPCSim (CB-UPCSim)," *Computers*, vol. 10, no. 10, Oct. 2021, https://doi.org/10.3390/computers10100123.

[24] Y. Yan, "The Overview of Elliptic Curve Cryptography (ECC)," *Journal of Physics: Conference Series*, vol. 2386, no. 1, Sept. 2022, Art. no. 012019, https://doi.org/10.1088/1742-6596/2386/1/012019.

[25] G. Narsimha and C. Sampath Reddy, "Secure Optimized Routing and Data Transmission in Wireless Sensor Networks with Elliptic Curve Cryptography," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 4, Aug. 2022, https://doi.org/10.22266/ijies2022.0831.26.

[26] S. K. Wagle, A. A. Bazilraj, and K. P. Ray, "Energy efficient security solution for attacks on Wireless Sensor Networks," in *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, Sept. 2021, pp. 313–318, https://doi.org/10.1109/ACCESS51619.2021.9563325.

[27] S. Pfeiffer and N. Tihanyi, "D(HE)at: A Practical Denial-of-Service Attack on the Finite Field Diffie–Hellman Key Exchange," *IEEE Access*, vol. 12, pp. 957–980, 2024, https://doi.org/10.1109/ACCESS.2023.3347422.

[28] Y. Xie *et al.*, "A Dual-Core High-Performance Processor for Elliptic Curve Cryptography in GF(p) Over Generic Weierstrass Curves," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 11, pp. 4523–4527, Aug. 2022, https://doi.org/10.1109/TCSII.2022.3194019.

[29] D. Lestari, I. D. Wahyono, and I. Fadlika, "IoT based Electrical Energy Consumption Monitoring System Prototype: Case study in G4 Building Universitas Negeri Malang," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Aug. 2017, pp. 342–347, https://doi.org/10.1109/SIET.2017.8304161.