STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, April 2026, pp 2928–2946.
Published online in International Academic Press (www.IAPress.org)

Optimized Ambiguous-Key ECC Signatures for Lightweight and Secure IoT Systems

Mohamed El Ouafi[1,*], Kaoutar Lamrini Uahabi[1], Abderrahim Aslimani[2], Abderrahim Zannou[3]

[1]*Department of Mathematics, Pluridisciplinary Faculty, Mohammed First University, Laboratory of Applied Mathematics and Information Systems, Selouane, Nador, Morocco*

[2]*Department of Mathematics, Pluridisciplinary Faculty, Mohammed First University, Laboratory of Applied Mathematics of the Oriental, BP 300-62700, Selouane, Nador, Morocco*

[3]*ERCI2A, FSTH, Abdelmalek Essaadi University, Tetouan, Morocco*

Abstract In IoT constrained environments, traditional digital signatures struggle to acheive an effective balance between security, compactness, and computational efficiency. To overcome these constraints, we propose a lightweight elliptic-curve signature scheme based on a dual-component private key (x, Q_1) and two auxiliary commitments, inspired by the Schnorr structure. The design introduces structural ambiguity in the secret key, increasing resistance to key-recovery attacks while maintaining a lightweight and fast signature process. Experimental evaluation on NIST-standardized elliptic curves shows competitive performance: key generation ranges from 11.6 ms to 232.1 ms, signing from 17.9 ms to 245.4 ms, and verification from 22.5 ms to 258.0 ms, with energy consumption below 2.8 μJ. The results confirm that the proposed scheme offers an effective a balanced compromise between compactness, runtime efficiency, energy usage, memory requirements, and practical security guarantees, making it suitable for distributed architectures and resource-constrained IoT devices.

Keywords Digital signatures, elliptic curves, Hidden Shift Problem, discrete logarithm problem , IoT.

DOI: 10.19139/soic-2310-5070-3280

1. Introduction

In today's digital era, given the growing dependence of modern life on the Internet and digital communications, the need for secure methods for authentication and data protection has become essential [1]. This need is particularly important with the expansion of the Internet of Things (IoT), electronic services, and the increase in cyber threats and fraud techniques [2, 3]. Beyond security concerns, large-scale IoT deployments must also ensure efficient data collection and routing in constrained networks, where poor data-flow management can increase latency, energy consumption, and destabilize network operation; this motivates the design of optimization mechanisms specifically tailored to constrained IoT environments [4]. Moreover, IoT-enabled digital services increasingly support integrity-critical applications that require robust authentication and verifiable traceability, such as blockchain-based e-voting systems, as well as digital signature mechanisms for securing electronic transactions [5].

Among the most important tools for securing electronic transactions we find digital signatures which ensure the integrity authentication and non-repudiation of digital data [6] in many areas especially for secure authentication between IoT devices and to ensure that the data has not been modified and that the sender is an authorized entity [2].

These signatures rely on several cryptographic aspects in particular Elliptic Curve Cryptography (ECC) which is considered a major advance to improve the efficiency and security of digital signatures compared to classical

*Correspondence to: Mohamed El Ouafi (Email: mohamed.elouafi10@ump.ac.ma). Department of Mathematics, Pluridisciplinary Faculty, Mohammed First University, Selouane, Nador, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

systems such as Schnorr [7] ElGamal [8] and RSA [9] based on cyclic groups of numbers the ECC environment provides the same security level as cyclic-group systems but with smaller keys saving memory and computation.

Common ECC schemes include ECDSA EdDSA and Bulletproofs [10, 11, 12] which are also widely used in security applications especially in cryptocurrencies like Bitcoin Ethereum Monero Solana and Algorand [13, 14, 15, 16, 17] because of their speed and efficiency these systems rely on standardized signatures such as ECDSA and EdDSA using curves like curve25519 [18] and secp256k1 [19], Moreover, recent *lightweight cryptography for IoT* research proposes schemes and system designs explicitly tailored to constrained devices, aiming to reduce computation and communication costs while preserving practical security; representative examples include ESEM, LRSHA/HASES, and LiteQSign [20, 21, 22].

However even if these schemes are efficient they still present several technical challenges that affect performance and security one challenge comes from side-channel attacks [23] where private keys can be compromised by analyzing system behavior and generated signatures [24] this requires nonces to be generated following strict rules to avoid repetition and some operations need heavy computation which can slow the system especially on devices with limited resources such as mobile phones or IoT nodes especially with the rapid development of quantum computing which requires a deep re-evaluation of classical cryptographic primitives because Shor's algorithm [25] breaks the security of systems based on the elliptic curve discrete logarithm problem (ECDLP) since a sufficiently large quantum computer could solve ECDLP in polynomial time as shown by recent analyses which indicate that a 256-bit curve such as **Secp256k1** used in Bitcoin could become vulnerable with a few thousand stabilized qubits [26] this threat has encouraged research into new quantum-resistant primitives.

Among alternative approaches schemes based on lattice problems such as LWE (Learning With Errors) [27] and SIS (Short Integer Solution) [28] offer strong security even against known quantum algorithms [29] however these constructions suffer from important limitations large key and signature sizes high computational cost and high memory usage which make them less suitable for resource-constrained environments such as IoT or high-throughput blockchains in contrast elliptic curve-based schemes remain lightweight fast and simple to implement but their resistance is weakened by Shor therefore a compromise is needed keeping the compactness and speed of elliptic systems while avoiding vulnerabilities to quantum computers.

In this context the *Hidden Shift Problem* (HSP) [30, 31] is a promising approach unlike ECDLP the HSP in non-abelian groups has no known quantum algorithm running in polynomial time the best attacks rely on subexponential algorithms such as Kuperberg's [32] later improved by Regev [33] These algorithms have complexity of order $2^{O(\sqrt{\log N})}$, which is much more costly than Shor's polynomial attacks, providing a notable security advantage. Moreover, several works link HSP to hard lattice problems, such as the *Shortest Vector Problem* (SVP), considered quantum-resistant [34]. This relation strengthens HSP as a robust cryptographic foundation.

In this framework, we propose a classical digital signature scheme inspired by the Schnorr structure [7]. Our construction is conceptually based on two ideas from known hard problems: the *Hidden Shift Problem* (HSP) and the *Hidden Number Problem* (HNP) [35]. Although the problem we introduce does not directly match the classical formulations of HSP or HNP, it is inspired by them to create structural ambiguity around the secret key, by combining a secret scalar with an additional elliptic point, making key recovery more complex. We emphasize that the resulting scheme is classical and does not offer post-quantum security..

The main idea is to replace the classical discrete logarithm problem (ECDLP) with an additive decomposition variant in an elliptic group. This approach keeps the lightness of Schnorr-like schemes while increasing cryptanalytic difficulty.

The underlying mathematical problem can be formulated as follows: given a generator G of an elliptic subgroup and a pair $(G, xG + Q)$, recover the secret value x and the associated point Q, knowing that Q is a random independent point [36]. This formulation reduces secret recovery to a sub-determined system with multiple solutions, for which the correct solution cannot be uniquely determined.

In our construction, the private key consists of a secret integer x and an additional elliptic point Q, denoted (x, Q). This dual component increases cryptanalytic difficulty: simultaneously extracting a secret scalar and a masked point is considered significantly more resistant than a simple ECDLP. This design aims to strengthen resistance against brute-force attacks while keeping a lightweight structure similar to ECDSA and Schnorr.

The proposed scheme aims to reinforce existing Schnorr or ECDSA-like signature schemes using the already defined problem on elliptic curve groups. This design aims to maintain the same security level offered by Schnorr and ECDSA, while reducing the curve size needed to guarantee this level of protection. The addition of an extra structural component to the secret key — combining a secret scalar with an independent elliptic point — makes direct key recovery rely on a sub-determined system, increasing analysis difficulty and limiting the effectiveness of attacks based on correlations among multiple signatures. Thus, extracting the public key via brute force or learning techniques from signatures becomes much harder compared to the standard Schnorr structure.

Section 2 presents an overview of related work. Section 3 describes the framework of our proposed signature scheme. Section 4 is devoted to security analysis with proof, and Section 5 illustrates simulations and results. Finally, Section 6 concludes the paper and discusses future perspectives.

2. Related works

Classical signature schemes are the foundation of modern digital authentication systems. Most of them rely on old and well-studied mathematical assumptions, mainly the discrete logarithm problem in cyclic groups or on elliptic-curve groups, and also the integer factorization problem. The most representative schemes include RSA, DSA, Schnorr, ECDSA, and EdDSA. Among these, the ECDSA and Schnorr signature schemes have a special historical importance, as they are based on the discrete logarithm problem in a large cyclic group.For example, in the Schnorr scheme, the signer chooses a random value k, computes the commitment $R = g^k$, then generates the challenge $e = H(R\|m)$, and finally computes the response $s = k - xe \bmod q$, where x is the private key. Verification checks whether $g^s y^e$ correctly reconstructs the public commitment. The strength of this protocol comes from its simplicity, elegance, and strong security guarantees in the random-oracle model.

Over time, elliptic curves led to more efficient variants. ECDSA, used for example in Bitcoin and many digital infrastructures, replaces modular exponentiation with elliptic-curve point multiplication. For a message m, the signer generates a random k, computes $R = kG$, then $r = x(R)$, and finally $s = k^{-1}(H(m) + dr) \bmod n$, forming the signature (r, s). Verification uses affine recombinations $u_1G + u_2Q$ and checks that the resulting x-coordinate matches r. ECDSA is extremely compact and efficient, but its security strongly depends on the uniqueness of the nonce: any reuse of k immediately compromises the private key. It is also vulnerable to Shor's quantum algorithm, which breaks the discrete logarithm problem on elliptic curves in polynomial time.

EdDSA, whose most widely used version is ed25519, provides a modern alternative using Edwards curves. It adopts a deterministic nonce derived through hashing, reducing implementation risks. Thanks to the geometric properties of Edwards curves, EdDSA offers fast signatures, simple implementations, and resistance to timing attacks, although it remains vulnerable to quantum attacks like ECDSA.

The table 1 presents an overview of the main classical signature schemes, describing their underlying hardness assumptions and the strengths and weaknesses of each scheme.

Scheme	Underlying problem	Strengths	Weaknesses
Schnorr	Discrete logarithm (cyclic groups or EC)	Simple security; elegant proof; short signatures	Not resistant to Shor; not optimal on elliptic curves
ECDSA	Elliptic-curve discrete logarithm (ECDLP)	Very compact; very efficient; widely deployed	Vulnerable to Shor; sensitive to nonce reuse
EdDSA	Discrete logarithm (Edwards curves)	Deterministic nonce; fast; secure implementations	Vulnerable to Shor; key size similar to ECDSA
RSA	Integer factorization	Historical standard; conceptually robust	Long signatures; inefficient on small devices; vulnerable to Shor

Table 1. Main classical signature schemes and general comparison

Key sizes for private and public keys are essential when evaluating the efficiency of a signature scheme. Table 2 shows the typical key sizes. Algorithms based on factorization (such as RSA) require very large keys to reach a given security level, while elliptic-curve-based schemes (ECDSA, EdDSA, Schnorr-EC) provide comparable security with much smaller keys.

Scheme	**Private key**	**Public key**
Schnorr (modular)	256 bits	3072 bits
Schnorr-EC (P-256)	256 bits	512 bits
ECDSA (P-256)	256 bits	512 bits
EdDSA (Ed25519)	256 bits	256 bits (compressed)
RSA (3072 bits)	3072 bits	3072 bits

Table 2. Typical key sizes for about 128-bit security.

Post-quantum signature schemes cover several distinct mathematical constructions. Each family has advantages and drawbacks in terms of security, size (public keys, private keys, signatures), and performance.

Lattice-based schemes are currently the most advanced family. They rely on problems such as LWE, SIS, or SVP, believed to remain hard even for quantum adversaries. Falcon [37] and CRYSTALS-Dilithium [38] are the most mature candidates and have been selected by the National Institute of Standards and Technology (NIST) [39] for standardization. These schemes offer a good compromise between efficiency, size, and security.

Hash-based signatures, with SPHINCS+ [40] as the most representative example, rely solely on the security of hash functions (collision and preimage resistance). This single dependency makes them conceptually simple and robust, but at the cost of significantly larger signatures compared to lattice-based or ECC schemes.

Code-based schemes, exemplified by Classic McEliece [41], have demonstrated strong robustness for decades, but suffer from extremely large public keys. Multivariate schemes (e.g., Rainbow) [42] rely on solving multivariate polynomial systems and can provide fast signatures, but recent cryptanalysis revealed vulnerabilities in some variants. Finally, isogeny-based schemes originally offered very compact keys, but recent attacks have compromised the security of several constructions.

Family	**Examples**	**Strengths**	**Limitations**
Lattice-based	Dilithium, Falcon	Efficient; strong security; NIST-selected	Larger keys or signatures than ECC
Hash-based	SPHINCS+	Very strong security; depends only on hashing	Large signatures; higher computation cost
Code-based	Classic McEliece	Very robust for decades	Extremely large public keys
Multivariate	Rainbow	Fast signatures; simple structure	Recent cryptanalytic weaknesses
Isogeny-based	SIKE	Compact keys; elegant mathematics	Security broken by recent attacks

Table 3. Overview of the main post-quantum signature families.

Table 3 summarizes the main families of post-quantum signature schemes and highlights their distinguishing features. Lattice-based schemes (*Dilithium, Falcon*) are considered the most promising, offering a strong balance between security and efficiency, as confirmed by their selection by the NIST, although their keys are larger than those of classical elliptic-curve schemes (ECC). Hash-based signatures (*SPHINCS+*) provide very strong security since they rely only on hash functions, but produce large signatures. Code-based schemes (*Classic McEliece*) are historically robust, but penalized by huge public keys.

Key and signature sizes, as shown in Table 4, vary greatly depending on the family. Lattice-based schemes generally offer a good balance (keys and signatures in the kilobyte range), while hash-based schemes produce very large signatures but keep small public keys. Code-based and multivariate schemes tend to have very large public or private keys, and isogeny-based constructions (when considered secure) offered interesting compactness.

Family	Example	Public key	Private key	Signature
Lattice-based	Dilithium-III	∼1.5 kB	∼2.0 kB	∼2.7 kB
	Falcon-512	∼0.9 kB	∼1.3 kB	∼0.66 kB
Hash-based	SPHINCS+-128	∼32 B	∼64 B	8–30 kB
Code-based	Classic McEliece	200–500 kB	5–10 kB	∼1 kB
Multivariate	Rainbow-I	100–150 kB	50–100 kB	∼66 B
Isogeny-based	SIKE (former)	0.2–0.4 kB	0.2–0.4 kB	∼0.2 kB

Table 4. Typical key and signature sizes for different post-quantum families.

The transition to post-quantum signature schemes therefore requires balancing security, compactness, and performance. Standardization efforts (notably those of NIST) take these trade-offs into account and now guide deployment choices to progressively replace classical signatures in modern protocols.

3. Our proposal

In this section, we present our proposed signature scheme over elliptic curve groups, designed to enhance security while preserving the classical computational assumptions

3.1. Key Generation

The parameters required for key generation are: p, a prime number defining the finite field $\mathbb{F}_p$; a, b, the coefficients of the elliptic curve E over $\mathbb{F}_p$; and G **and** n, where G is a generator point of a subgroup $\langle G\rangle_n \subseteq E(\mathbb{F}_p)$ of prime order n.

The key generation process starts by randomly choosing an integer

$$x \in \{1, \ldots, n-1\}, \tag{1}$$

which is the main secret of the signer. Next, a random point

$$Q_1 \in E(\mathbb{F}_p) \tag{2}$$

is chosen to introduce a random additive component. These two values are an essential part of the hidden structure used by our scheme. The public key is then defined as

$$P = xG + Q_1, \tag{3}$$

where xG is the scalar multiplication of G by the secret x. The private key is the pair (x, Q_1), and the public key is P. The full procedure is shown in Algorithm 1.

Algorithm 1 KeyGen()

Require: Curve parameters: p, (a, b), generator G of prime order n.
Ensure: Public parameters (p, a, b, n, G), public key P, private key (x, Q_1).

1: Choose $x \xleftarrow{\$} \{1, \ldots, n-1\}$.
2: Choose a random point $Q_1 \xleftarrow{\$} E(\mathbb{F}_p)$.
3: Compute the public key $P \leftarrow xG + Q_1$.
4: **Output:** public parameters (p, a, b, n, G), public key P, private key (x, Q_1).

3.2. Signing Process

The signing process, shown in Algorithm 2, takes the private key (x, Q_1) and message m as input. It starts by selecting two random integers $k_1, k_2 \in [1, n-1]$ and a random point $Q_2 \xleftarrow{\$} E(\mathbb{F}_p)$. Generate nonces using a

CSPRNG (or deterministically from (sk, m), RFC 6979-style). Then, it computes the nonce

$$A_1 = k_1 G - Q_2, \tag{4}$$

checks that $A_1 \neq \mathcal{O}$, extracts the x-coordinate of A_1, called x_{A_1}, and checks that x_{A_1} is invertible modulo n (i.e., $x_{A_1} \not\equiv 0 \pmod{n}$ and $\gcd(x_{A_1}, n) = 1$) before computing the hash:

$$e = H(x_{A_1} \| m), \tag{5}$$

where H is a cryptographic hash function. Next, compute

$$A_2 = k_2 G + e \cdot x_{A_1}^{-1} Q_1, \tag{6}$$

check $A_2 \neq \mathcal{O}$, and extract $x_{A_2} = \text{affix}_x(A_2) \bmod n$ to compute:

$$s = (e \cdot x + x_{A_2} - k_2 \cdot x_{A_1}) \bmod n. \tag{7}$$

Finally, the algorithm outputs the signature (s, A_1, A_2). Here, x_{A_i} denotes the scalar $x_{A_i} := \text{affix}_x(A_i) \bmod n \in \mathbb{Z}_n$, its x-coordinate. When a scalar in $\mathbb{Z}_n$ is required.

Algorithm 2 Sign()

Require: Private key (x, Q_1), message m.
Ensure: Signature (s, A_1, A_2).

1: Generate nonces using a CSPRNG (or deterministically from (sk, m), RFC 6979-style).
2: Choose $k_1, k_2 \xleftarrow{\$} \{1, \ldots, n-1\}$.
3: Choose a random point $Q_2 \xleftarrow{\$} E(\mathbb{F}_p)$.
4: Compute $A_1 \leftarrow k_1 G - Q_2$.
5: Check $A_1 \neq \mathcal{O}$ and compute $x_{A_1} := \text{affix}_x(A_1) \bmod n$.
6: **if** $x_{A_1} \equiv 0 \pmod{n}$ **or** $\gcd(x_{A_1}, n) \neq 1$ **then**
7: reject and restart with new k_1 and Q_2.
8: **end if**
9: Compute $e \leftarrow H(x_{A_1} \| m)$ (value in $\mathbb{Z}_n$).
10: Compute

$$A_2 \leftarrow k_2 G + e \cdot x_{A_1}^{-1} Q_1.$$

11: Check $A_2 \neq \mathcal{O}$ and compute $x_{A_2} := \text{affix}_x(A_2) \bmod n$.
12: Compute

$$s \leftarrow (e \cdot x + x_{A_2} - k_2 \cdot x_{A_1}) \bmod n.$$

13: **Output:** (s, A_1, A_2).

3.3. Signature Verification

The verification process, shown in Algorithm 3, takes the public key P, message m, and signature (s, A_1, A_2). It first checks that A_1, A_2 are valid points on $E(\mathbb{F}_p)$ and not $\mathcal{O}$, then computes:

$$R_1 = x_{A_1} \cdot A_2 \,, \quad R_2 = x_{A_2} \cdot G \,, \quad R_s = s \cdot G.$$

Then, it extracts x_{A_1} and computes the hash

$$e = H(x_{A_1} \| m).$$

The signature is valid if

$$R_2 + e \cdot P = R_1 + R_s. \tag{8}$$

If this holds, the signature is accepted; otherwise, it is rejected.

Algorithm 3 Verify()

Require: Public parameters (p, a, b, n, G), public key P, message m, signature (s, A_1, A_2).
Ensure: Accept / reject.

1: Check A_1, A_2 are valid points on $E(\mathbb{F}_p)$ and $\neq \mathcal{O}$.
2: Compute $x_{A_1} := \text{affix}_x(A_1) \bmod n$. Reject if $x_{A_1} \equiv 0 \pmod n$.
3: Compute $e \leftarrow H(x_{A_1} \| m)$.
4: Compute $x_{A_2} := \text{affix}_x(A_2) \bmod n$.
5: Compute points:

$$R_1 \leftarrow x_{A_1} \cdot A_2, \quad R_2 \leftarrow x_{A_2} \cdot G, \quad R_s \leftarrow s \cdot G.$$

6: **if** $R_2 + e \cdot P = R_1 + R_s$ **then**
7: Accept.
8: **else**
9: Reject.
10: **end if**

All scalar operations are done modulo n. When x_{A_i} is used as a scalar (or its inverse $x_{A_i}^{-1}$), we assume a canonical mapping from $\mathbb{F}_p$ to $\mathbb{Z}_n$ and that x_{A_1} is invertible modulo n. The scheme parameters are those defined in the key generation and signing algorithms.

It can be shown that every signature (s, A_1, A_2) produced by the signing algorithm satisfies the verification equation:

$$R_2 + e \cdot P = R_1 + R_s, \tag{9}$$

where

$$R_1 = x_{A_1} \cdot A_2 \,, \quad R_2 = x_{A_2} \cdot G \,, \quad R_s = s \cdot G \,, \quad e = H(x_{A_1} \| m).$$

Thus, any honest signature is always accepted by the verification algorithm.

Recall:

$$\begin{aligned} P &= xG + Q_1, \\ A_1 &= k_1 G - Q_2, \\ A_2 &= k_2 G + e\, x_{A_1}^{-1} Q_1, \\ s &= (e \cdot x + x_{A_2} - k_2 \cdot x_{A_1}) \bmod n. \end{aligned} \tag{10}$$

Compute the three terms R_1, R_2, R_s:

$$\begin{aligned} R_1 &= x_{A_1} \cdot A_2 = k_2 x_{A_1} \cdot G + e \cdot Q_1, \\ R_2 &= x_{A_2} \cdot G, \\ R_s &= s \cdot G = ex \cdot G - k_2 x_{A_1} \cdot G + x_{A_2} \cdot G. \end{aligned} \tag{11}$$

Adding $R_1 + R_s$ and simplifying:

$$R_1 + R_s = e\, Q_1 + ex\, G + x_{A_2} \cdot G = x_{A_2} \cdot G + e \cdot P, \tag{12}$$

which exactly matches the verification equation. Therefore, Verify accepts any honest signature.

4. Security proof

vIn this section, we prove the security of our scheme in the **EUF-CMA** model with the **ROM**. Before that, we formally define the underlying problem on which our signer is based and show its structural hardness.

4.1. Hardness of the Underlying Problem and Structural Resistance

We study in detail the intrinsic difficulty of the elliptic-curve problem that supports the structural security of our modified protocol. This analysis shows that an adversary cannot recover either the private key x or the additional components Q_1, Q_2, because the solutions are ambiguous and never unique.

Definition 4.1. Let $\mathbb{E}(\mathbb{F}_p)$ be an elliptic-curve group with p prime, and let $G \in \mathbb{E}(\mathbb{F}_p)$ be a generator of prime order n. Let $x \in \mathbb{F}_p^*$ and $Q_1 \in \mathbb{E}(\mathbb{F}_p)$, and define $P = xG + Q_1$. The problem is to find (x, Q_1) from (G, P).

Without any extra information about x or Q_1, this problem is at least as hard as the discrete logarithm problem (DLP). If $\#\mathbb{E}(\mathbb{F}_p) = n \times m$ with n and m prime, then

$$\mathbb{E}(\mathbb{F}_p) \cong \mathbb{Z}_n \times \mathbb{Z}_m \cong \langle G \rangle \times \langle G' \rangle. \tag{13}$$

Thus, $Q_1 = aG + bG'$ for some $a, b \in \mathbb{F}_p$, which gives

$$P = (x + a)G + bG'.$$

Recovering $(x + a)$ and b does not reveal the true pair (x, Q_1). For any $a' \in \mathbb{F}_p$, there exists $x' \in \mathbb{Z}_n$ such that $x' + a' = x + a$. Hence (x, Q_1) is never unique, and for every $t \in \mathbb{F}_p$:

$$P = x'G + Q_1' \quad \text{with} \quad x' = x + t,\ Q_1' = Q_1 - tG.$$

We show that solving the above decomposition problem is at least as hard as the elliptic curve discrete logarithm problem (ECDLP).

Let $\mathcal{A}$ be an algorithm that, given (G, P), outputs a valid pair (x, Q_1) such that $P = xG + Q_1$. We construct an algorithm $\mathcal{B}$ that solves the ECDLP.

Given an ECDLP instance (G, Y), where $Y = \alpha G = \alpha G + \mathcal{O}$ and the goal is to recover α, algorithm $\mathcal{B}$ runs $\mathcal{A}$ on input $(G, P = Y)$. By assumption, $\mathcal{A}$ outputs $(\alpha, \mathcal{O})$.

Therefore, $\alpha \equiv x + \beta \pmod{n}$ is a discrete logarithm of Y to the base G, which can be recovered from the output of $\mathcal{A}$. Hence, any algorithm that solves the decomposition problem can be used to solve the ECDLP.

It follows that the problem of recovering (x, Q_1) from (G, P) is at least as hard as the elliptic curve discrete logarithm problem.

Therefore, a direct attack on the public key, even by brute force, cannot extract the true private pair (x, Q_1). More importantly, the transcripts generated by our modified signer leak no useful information about (x, Q_1, Q_2) or about breaking the scheme. The transcripts are:

$$\begin{cases} P = xG + Q_1, \\ A_1 = k_1 G - Q_2, \\ A_2 = k_2 G + e\, x_{A_1}^{-1} Q_1, \\ s = ex + x_{A_2} - k_2 x_{A_1} \pmod{n}. \end{cases} \tag{14}$$

If an attacker tries exhaustive search to extract discrete logs in the basis $\langle G \rangle \times \langle G' \rangle$, the system becomes

$$\begin{cases} P = (x + t_1)G + t_1' G', \\ A_1 = (k_1 + t_2)G + t_2' G', \\ A_2 = (k_2 + c_1 t_1)G + (c_1 t_1')G'. \end{cases} \tag{15}$$

where $c_1 = e\,x_{A_1}^{-1}$ and $t_i, t_i' \in \mathbb{Z}_n$. Simplifying 14 gives the system 16:

$$\begin{cases} x + t_1 = q_1, \\ k_1 + t_2 = q_2, \\ k_2 + c_1 t_1 = q_3, \\ ex + x_{A_2} - k_2 x_{A_1} = q_4. \end{cases} \tag{16}$$

All values $q_1, q_2, q_3, q_4, e, x_{A_1}, x_{A_2}, c_1$ are known.

This system has five unknowns (x, k_1, k_2, t_1, t_2) and only four equations. Therefore, it has one degree of freedom and an entire family of solutions. For large n, finding the correct one becomes very hard.

Even with multiple signatures $(s^{(i)}, A_1^{(i)}, A_2^{(i)})$ for different messages $m^{(i)}$, every signature adds new unknowns $(k_1^{(i)}, k_2^{(i)}, t_1^{(i)}, t_2^{(i)})$ but only four new equations. The whole system remains underdetermined, which keeps the structural security intact.

In classical elliptic-curve signatures such as ECDSA on a 256-bit curve, the effective security is about 128 bits, because the best known attack on ECDLP is Pollard–Rho [43] which runs in $O(\sqrt{n})$, giving about 2^{128} operations for a 256-bit curve.

Our scheme is structurally different: the attacker must recover both x and t_1 at the same time. This adds an extra $O(n)$ search. So the underlying problem is strictly harder than the standard ECDLP.

Thus, while a classical scheme requires a 256-bit curve to reach 128-bit security, our scheme achieves the same security level using a 128-bit curve, making it much lighter and suitable for constrained IoT environments.

4.2. Resistance to Signature Forgery (EUF-CMA)

The goal of this section is to show that any attack able to produce a valid forgery in the EUF-CMA model with non-negligible probability would allow recovering the secret key x. Such a recovery would break the hardness assumption of the underlying problem. Conversely, if this problem is indeed hard, then no EUF-CMA forgery is possible, and the scheme is secure.

theorem 4.2. *Assume that the hash function H is modeled as a random oracle. Also assume that it is hard to recover x from the public point $P = xG + Q$. If a probabilistic polynomial-time adversary $\mathcal{A}$, making at most q_H hash queries, forges a signature with probability ϵ, then there exists an algorithm $\mathcal{B}$ that can extract the key x with probability at least*

$$\frac{\epsilon^2}{2q_H} - \mathrm{negl}(\lambda). \tag{17}$$

This would contradict the hardness assumption and implies that the scheme is EUF-CMA secure.

Proof

The algorithm $\mathcal{B}$ simulates the hash oracle and the signing oracle for the adversary $\mathcal{A}$. The hash $e = H(x_{A_1} \| m)$ is only defined after fixing the commitment A_1, which satisfies the condition needed for the forking lemma. This timing ensures that two separate executions of the hash oracle on the same transcript can produce different values e and e' without changing other transcript elements.

When $\mathcal{B}$ knows x, the EUF-CMA game is simulated exactly. A forgery occurring with probability ϵ leads, via the forking lemma, to two valid signatures on the same message m^* with the same commitment A_1^* but two different hashes $e \neq e'$. The two signature verification equations are then

$$s \equiv ex - k_1 x_P - k_2 x_{A_1} + x_{A_2} \pmod{n}, \tag{18}$$

$$s' \equiv e'x - k_1 x_P - k_2 x_{A_1} + x'_{A_2} \pmod{n}. \tag{19}$$

Subtracting them gives

$$s - s' \equiv (e - e')x + (x_{A_2} - x'_{A_2}) \pmod{n}. \tag{20}$$

The term $x_{A_2} - x'_{A_2}$ does not depend on x and is fully determined by public values generated during the forgery. Hence, the secret key can be isolated:

$$x \equiv \frac{(s - s') - (x_{A_2} - x'_{A_2})}{e - e'} \pmod{n}. \tag{21}$$

The forking lemma ensures that obtaining two consistent forgeries happens with probability at least $\epsilon^2/(2q_H)$, minus a negligible term.

If $\mathcal{B}$ does not know x and only has P, it simulates the real signer by programming the random oracle. For a signature request on message m, it randomly chooses k_1, k_2 and an auxiliary point Q_2, sets $A_1 = k_1 G - Q_2$, and picks a uniform $x_{A_2} \in \mathbb{Z}_n$. The hash oracle is programmed to define e so that the verification equation

$$x_{A_2} G + eP = x_P A_1 + x_{A_1} A_2 + sG \tag{22}$$

holds. This determines the correct s. The resulting distribution is indistinguishable from that of a real signer, except with negligible probability depending on the hash output size.

When a valid forgery on a new message occurs, applying the forking lemma yields two valid signatures with the same A_1 and the same pairs (k_1, k_2), but different hashes $e \neq e'$. The same algebraic manipulation then allows $\mathcal{B}$ to extract the secret key x. Losses due to programming the ROM remain negligible for a large enough hash space.

Hence, the existence of a non-negligible forgery would directly violate the hardness assumption of the underlying problem, which establishes the EUF-CMA security of the scheme. □

5. Implementation and experimental results

This section explains the details of the implementation of our digital signature scheme on the NIST-standardized elliptic curves and presents experimental results in terms of energy consumption and execution time.

5.1. Performance Evaluation

The implementation was done using the Python 3.12 `cryptography` library for elliptic curve cryptography on a personal computer with the following configuration: `Intel(R) Core(TM) i7-8565U @ 1.80GHz` processor, 16 GB RAM, and a 512 GB SSD. The operating system was `Ubuntu 22.04 LTS`.

Energy measurements were performed using a high precision power analyzer (`pyRAPL`) and execution times were measured using system timing functions (`time.perf_counter()` in Python) to ensure a reliable evaluation of performance and energy consumption

Listing 1 shows the console output during execution on all tested NIST curves which serves two purposes one is to confirm the correct operation of the scheme for each curve and the other is to show the performance variations in time and energy for different cryptographic parameter sizes

Listing 1: Energy and time comparison for different NIST curves

```
=== Curve: secp128r1 ===          | === Curve: secp128k1 ===
Keygen energy: [142395.0] nJ      | Keygen energy: [66772.0] nJ
Keygen time: 16.302 ms            | Keygen time: 11.604 ms
Signature energy: [137756.0] nJ   | Signature energy: [174499.0] nJ
Signature time: 17.886 ms         | Signature time: 21.410 ms
Verify energy: [171386.0] nJ      | Verify energy: [159850.0] nJ
Verify time: 22.537 ms            | Verify time: 22.752 ms
Valid signature: True             | Valid signature: True

=== Curve: secp192r1 ===          | === Curve: secp224r1 ===
```

```
Keygen energy: [182982.0] nJ      | Keygen energy: [205078.0] nJ
Keygen time: 27.303 ms            | Keygen time: 32.694 ms
Signature energy: [431456.0] nJ   | Signature energy: [477172.0] nJ
Signature time: 43.772 ms         | Signature time: 50.446 ms
Verify energy: [438658.0] nJ      | Verify energy: [478026.0] nJ
Verify time: 49.578 ms            | Verify time: 43.396 ms
Valid signature: True             | Valid signature: True

=== Curve: secp256r1 ===          | === Curve: secp384r1 ===
Keygen energy: [345153.0] nJ      | Keygen energy: [886777.0] nJ
Keygen time: 38.895 ms            | Keygen time: 110.649 ms
Signature energy: [595701.0] nJ   | Signature energy: [1407589.0] nJ
Signature time: 57.168 ms         | Signature time: 123.560 ms
Verify energy: [720762.0] nJ      | Verify energy: [1533016.0] nJ
Verify time: 116.930 ms           | Verify time: 146.067 ms
Valid signature: True             | Valid signature: True

=== Curve: secp521r1 ===          |
Keygen energy: [1684505.0] nJ     |
Keygen time: 232.127 ms           |
Signature energy: [2759942.0] nJ  |
Signature time: 245.430 ms        |
Verify energy: [2737054.0] nJ     |
Verify time: 257.989 ms           |
Valid signature: True             |
```

Tables 5 and 6 summarize respectively the execution times in seconds and the energy consumption in nanojoules measured for the signature scheme operations on different tested NIST curves these data show in detail the general trend discussed previously namely that computational and energy costs increase with the growth of the computing environment for example Table 5 illustrates how execution times increase for key generation (*KeyGen*) which grows more than 14 times between the smallest curve (`secp128k1` at 0.0116 s) and the largest (`secp521r1` at 0.2321 s) and the signing operation increases about 11.5 times between `secp128k1` (0.0214 s) and `secp521r1` (0.2454 s) then verification which experienced the highest increase in particular from 0.1169 s on `secp256r1` to 0.2580 s on `secp521r1`.

The energy measurements presented in Table 6 show that energy consumption follows a pattern similar to execution times since there is a strong correlation between these two metrics for example the energy required for key generation goes from 66.8 µJ for `secp128k1` to 1.68 mJ for `secp521r1` which is about an increase of 25 times this marked growth is also found for signing where energy consumption increases by a factor of about 16 between `secp128r1` (137.8 µJ) and `secp521r1` (2.76 mJ) while the gap between signing and verification is also observed in energy consumption and not only in time especially for large curves.The energy impact is significant on high security curves such as (`secp384r1` and `secp521r1`) whose consumption consistently exceeds one millijoule a critical threshold for many resource constrained embedded applications fortunately this observation shows the value of our scheme which can operate with small curves while still maintaining strong security.

NIST Curve	KeyGen (s)	Signature (s)	Verification (s)
secp128r1	0.016302	0.017886	0.022537
secp128k1	0.011604	0.021410	0.022752
secp192r1	0.027303	0.043772	0.049578
secp224r1	0.032694	0.050446	0.043396
secp256r1	0.038895	0.057168	0.116930
secp384r1	0.110649	0.123560	0.146067
secp521r1	0.232127	0.245430	0.257989

Table 5. Execution time of cryptographic operations (in seconds)

NIST Curve	KeyGen (nJ)	Signature (nJ)	Verified (nJ)
secp128r1	142 395	137 756	171 386
secp128k1	66 772	174 499	159 850
secp192r1	182 982	431 456	438 658
secp224r1	205 078	477 172	478 026
secp256r1	345 153	595 701	720 762
secp384r1	886 777	1 407 589	1 533 016
secp521r1	1 684 505	2 759 942	2 737 054

Table 6. Energy consumption of cryptographic operations (in nanojoules)

Figure 1 shows the execution times for the three signature operations on NIST standardized elliptic curves it shows the proportional relationship between curve size and computational cost small curves like `secp128r1` and `secp128k1` have very low latencies of 25 ms while intermediate curves (`secp192r1` and `secp224r1`) show a noticeable increase in execution times this is more marked for high security level curves reflecting the arithmetic complexity as curves grow signing with `secp256r1` for example remains moderate but verification takes more than 110 ms however large curves like `secp384r1` and `secp521r1` show the strongest increase with key generation taking 110 ms and 230 ms respectively and verification reaching nearly 260 ms for `secp521r1`.

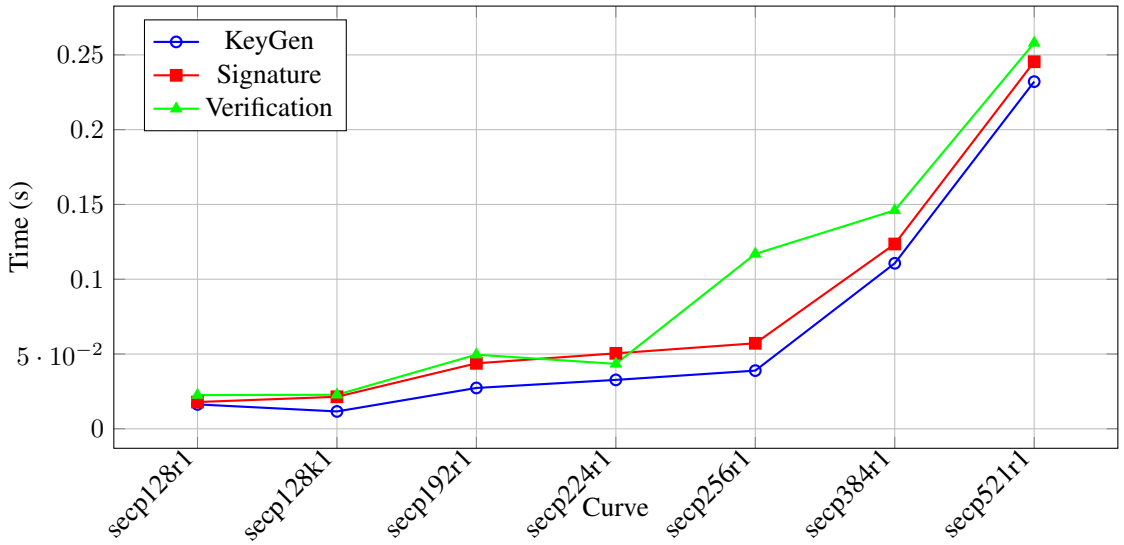


Figure 1. Execution time trends for different elliptic curves

Similarly Figure 2 shows the energy consumption for the same operations on NIST curves the results follow a pattern similar to execution time increasing systematically with curve size small curves (`secp128r1` and `secp128k1`) consume less than 200 µJ per operation making them very suitable for constrained environments with reasonable security while intermediate curves increase significantly in the same way as the largest curves (`secp384r1` and `secp521r1`) which have energy consumption well over one millijoule for most operations. These experimental results show that our scheme works efficiently on all tested NIST curves the validity of signatures is ensured for each curve showing the correctness of the scheme even with the predictable increase in computational resources with larger curves it is important to choose a curve according to the specific requirements of the application.

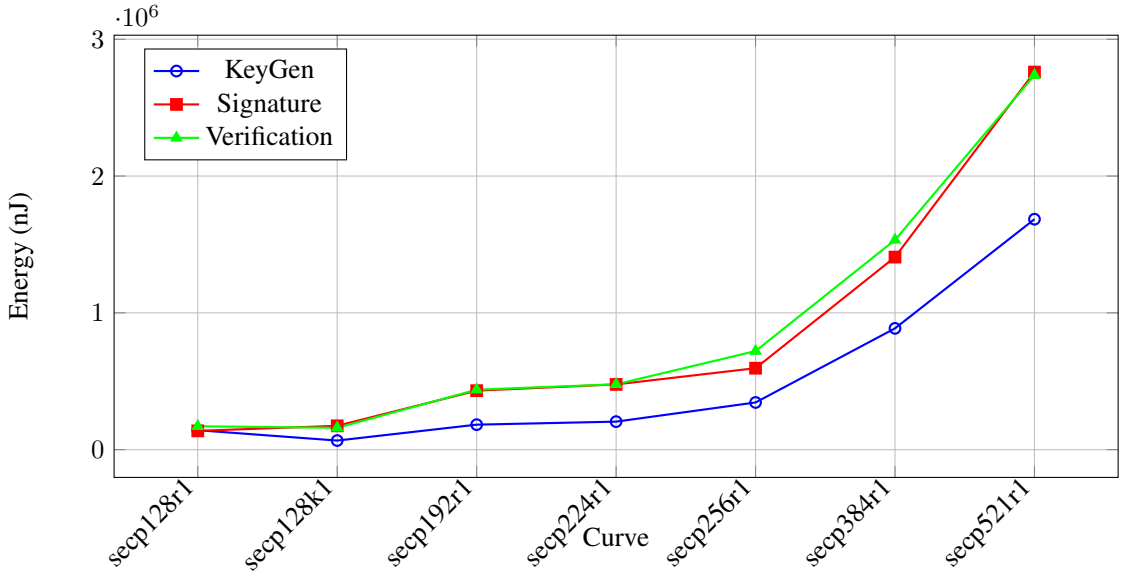


Figure 2. Energy consumption trends for different elliptic curves

5.2. Comparative Simulation

Listing 2 presents the raw measurements of execution time energy consumption private and public key sizes and signature sizes for our scheme as well as for ECDSA Schnorr Falcon and Dilithium-2 for a 128-bit security level as these measurements were obtained using the same experimental parameters and the same workstation described in subsection 5.1. We highlight that all these results come from Python implementations without hardware specific optimization and we specify that for Falcon and Dilithium-2 we used existing ready made libraries while for ECDSA Schnorr and our own scheme we built and implemented the entire scheme manually following the same experimental principles.

Listing 2: Key sizes, signature sizes, execution time and energy (ms / nJ) for all schemes

```
1
2
3  === Falcon ===
4
5  KeyGen: 30.350 ms, Energy = 144226000 nJ
6  Private key: 1281 bytes, Public key: 897 bytes
7  Sign: 7.974 ms, Energy = 32348000 nJ, Sign size = 654 bytes
8  Verify: 0.092 ms, Energy = 842400 nJ
9  Signature valid: True
10
11 === Dilithium-2 ===
12
13 KeyGen: 0.3748 ms, Energy = 3112800 nJ
14 Private key: 2560 bytes, Public key: 1312 bytes
15 Sign: 1.4374 ms, Energy = 7702600 nJ, Sign size = 2420 bytes
16 Verify: 0.2947 ms, Energy = 2087400 nJ
17 Signature valid: True
18
19 === ECDSA P-256 ===
```

```
KeyGen: 73.2992 ms, Energy = 767820.0 nJ
Private key: 32 bytes, Public key: 64 bytes
Sign: 87.8141 ms, Energy = 824461.0 nJ, Sign size = 64 bytes
Verify: 115.0637 ms, Energy = 1303280.0 nJ
Signature valid: True

=== Schnorr ===

KeyGen: 52.705 ms, Energy = 333800.0 nJ
Private key: 32 bytes, Public key: 64 bytes
Sign: 38.302 ms, Energy = 332092.0 nJ, Sign size = 64 bytes
Verify: 58.312 ms, Energy = 636473.0 nJ
Signature valid: True

=== Our Scheme on secp128r1 ===

KeyGen: 16.409 ms, Energy = 71960.0 nJ
Private key: ~32 bytes, Public key: ~32 bytes
Sign: 27.087 ms, Energy = 197631.0 nJ, Sign size = 50 bytes
Verify: 25.449 ms, Energy = 160644.0 nJ
Signature valid: True
```

Table 7. Comparison of execution times and key/signature sizes for each scheme

Table 7 (a): Execution times (ms)

Scheme	KeyGen	Sign	Verify
Falcon	30.350	7.974	0.092
Dilithium-2	0.3748	1.4374	0.2947
ECDSA P-256	73.2992	87.8141	115.0637
Schnorr	52.705	38.302	58.312
Our Scheme	16.409	27.087	25.449

Table 7 (b): Key and signature sizes (bytes)

Scheme	Private key	Public key	Signature
Falcon	1281	897	654
Dilithium-2	2560	1312	2420
ECDSA P-256	32	64	64
Schnorr	32	64	64
Our Scheme	~32	~32	50

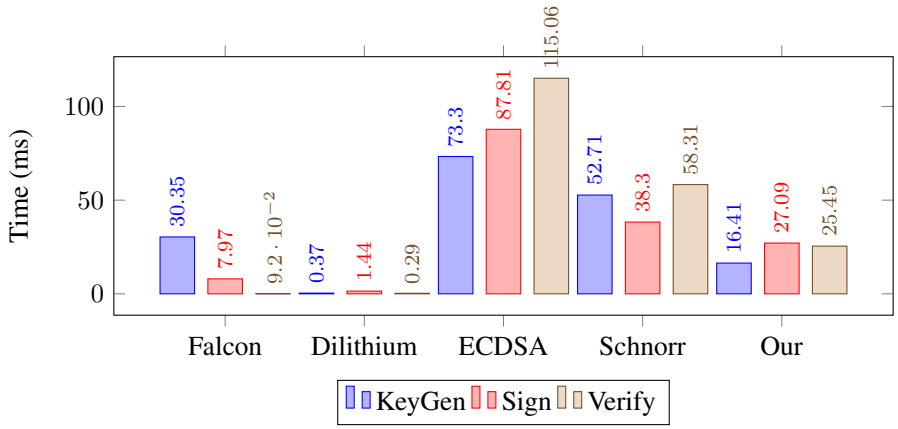
Table 7 summarizes the execution times, as well as the key and signature sizes for each studied scheme, such that it can be observed that Dilithium-2 is extremely fast for key generation and signing, with respective times of 0.375 ms and 1.437 ms, but this speed is accompanied by large key and signature sizes, with private keys of 2560 bytes and signatures of 2420 bytes, which makes it less suitable for resource-constrained environments, while ECDSA P-256 and Schnorr P-256, although standardized and having compact keys (32 bytes for private keys and 64 bytes for public keys), present high signing and verification times, reaching 115 ms for ECDSA verification and 58 ms for Schnorr verification, whereas these long times can limit their use in applications requiring low latency, then we find the Falcon-512 scheme which presents a different compromise such that it offers fast signing (7.97 ms) with intermediate key sizes (1281 bytes for the private key and 897 bytes for the public key), but it is costly in memory and energy, which can be problematic for devices with limited resources, and finally our proposed scheme, based on secp128r1, combines speed and efficiency, where key generation requires 16.41 ms, signing 27.09 ms, and verification 25.45 ms, and the keys and signatures remain compact, with about 64 bytes for keys and

50 bytes for the signature. This combination of performance and lightness makes our scheme particularly suitable for applications requiring both efficiency and low resource consumption, such as IoT and embedded systems.

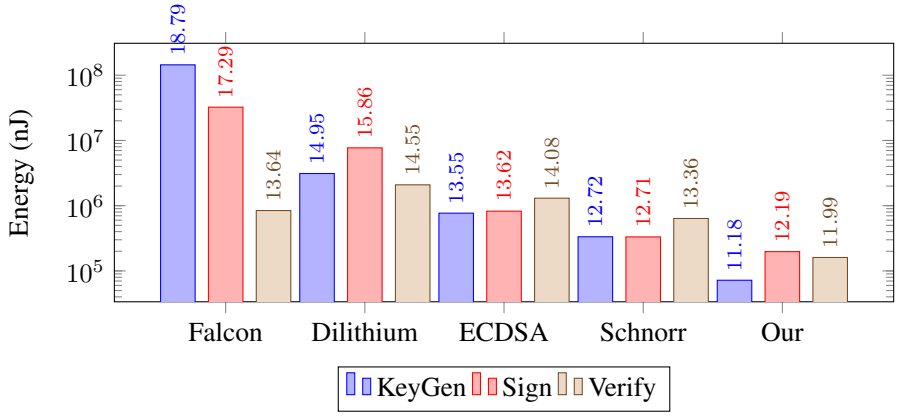
Table 8. Comparison of energy consumption (nJ) for each scheme

Scheme	KeyGen (nJ)	Sign (nJ)	Verify (nJ)
Falcon	144226000	32348000	842400
Dilithium-2	3112800	7702600	2087400
ECDSA P-256	767820	824461	1303280
Schnorr	333800	332092	636473
Our Scheme	71960	197631	160644

Table 8 compares energy consumption. Falcon is very energy-demanding, especially for key generation (144.226 mJ) and signing (32.348 mJ). Dilithium-2 is relatively efficient (3.1128 mJ for key generation and 7.7026 mJ for signing), while ECDSA P-256 and Schnorr use less energy but take longer to compute. Our scheme is the most balanced, with low energy consumption for all operations (0.07196 mJ for key generation, 0.197631 mJ for signing), while keeping reasonable key and signature sizes. Combining time and energy analysis clearly identifies the schemes best suited to performance- and energy-constrained environments.



(a) Execution time (ms)



(b) Energy consumption (log scale)

Figure 3. Comparison of execution time (a) and energy (b) for all schemes, with values displayed on each bar.

Figure 3 gives a visual comparison of the signature schemes in terms of execution time and energy. Subfigure (a) shows that Dilithium-2 is by far the fastest for key generation and signing, while ECDSA P-256 and Schnorr have higher computation times. Falcon allows fast signing but has costly key generation. Our scheme is relatively fast due to the smaller curve, keeping reasonable execution times and intermediate signature size. Subfigure (b), on a logarithmic scale, highlights that Falcon consumes the most energy, especially for key generation and signing, while Dilithium-2 stays moderate. ECDSA P-256 and Schnorr consume less energy but have longer execution times. Our scheme presents a balanced profile: faster and less energy-demanding than ECDSA and Schnorr, with acceptable execution times, making it suitable for constrained IoT environments.

5.3. Evaluation on Resource-Constrained Devices (Raspberry Pi)

We consider three signature schemes based on short Weierstrass elliptic curves. The ECDSA and EC Schnorr schemes use the **secp160r1** curve (cofactor $h = 1$), while our proposed scheme uses the **secp112r2** curve (cofactor $h = 4$), as described in Section 3. The target device is a **Raspberry Pi 4** (Cortex-A72, 1.5 GHz) simulated using `QEMU`. The metrics considered are the average execution time and the estimated energy consumption.

Each operation (key generation, signing, verification) was repeated $N = 500$ times to obtain stable averages. The energy is estimated by multiplying the CPU time by 20 J/s, which serves as a proxy for energy consumption on this device. Table 9 summarizes the results (time in ms, energy in µJ).

Scheme	Phase	Average Time (ms)	Average CPU (s)	Approx. Energy (µJ)
ECDSA	KeyGen	238.737	0.23860607	4772121.46
Schnorr	KeyGen	247.055	0.24671619	4934323.79
MySchema	KeyGen	206.271	0.20616491	4123298.15
ECDSA	Sign	234.758	0.23472250	4694450.02
Schnorr	Sign	234.280	0.23419917	4683983.41
MySchema	Sign	386.813	0.38640707	7728141.31
ECDSA	Verify	504.422	0.50409429	10081885.80
Schnorr	Verify	478.543	0.47842604	9568520.80
MySchema	Verify	395.974	0.39583288	7916657.67

Table 9. Average execution time and estimated energy for each scheme on a Raspberry Pi 4 simulated via QEMU, using the **secp112r2** curve (cofactor $h = 4$ for MySchema, **secp160** for ECDSA and Schnorr).

The results show that the proposed scheme is slower than ECDSA and Schnorr during the signing phase, with a time of approximately 389 ms, but faster for key generation and verification, with times of about 206 ms and 396 ms respectively, while consuming proportionally less energy, i.e., 4,123,298.15 µJ and 7,916,657.67 µJ. This slowdown during signing is expected: the scheme selects two random numbers and a point, then performs a modular inversion before computing the signature. However, these operations can be optimized. Nevertheless, all operations remain fully feasible on a simulated Raspberry Pi 4.

Figure 4 shows that the key generation times for ECDSA and Schnorr are similar, around 239 ms and 247 ms respectively, while the proposed scheme requires approximately 206 ms, slightly faster than the classical schemes in this case. This reflects the combined complexity of standard scalar multiplication operations along with additional points to ensure enhanced security.

For the signing phase, ECDSA and Schnorr perform similarly, around 234 ms, while the proposed scheme reaches 387 ms, approximately 1.6 times slower. This difference is due to the computation of multiple points and the application of several scalar operations on these points, increasing the computational cost.

For verification, ECDSA and Schnorr remain close, around 504 ms and 479 ms respectively, while the proposed scheme uses approximately 396 ms. Verification is therefore slightly faster than the classical schemes in this context and remains fully feasible on a simulated Raspberry Pi.

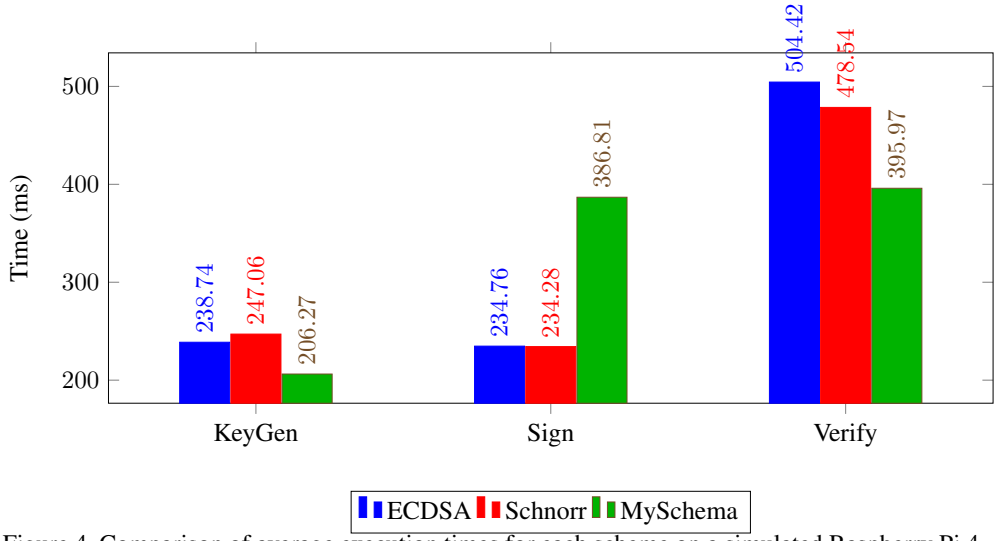


Figure 4. Comparison of average execution times for each scheme on a simulated Raspberry Pi 4.

The figure 5 complements this analysis by showing the approximate energy consumption for each operation. The proposed scheme consumes less energy for key generation (about 4.12 mJ) compared to ECDSA and Schnorr (4.77 mJ and 4.93 mJ, respectively). For signing, the energy cost of the proposed scheme increases (7.73 mJ), reflecting the additional operations required, but remains comparable to the values of Schnorr and ECDSA. Finally, for verification, the proposed scheme is slightly more energy-efficient (7.92 mJ) than the classical schemes, confirming that all operations remain feasible and efficient on embedded platforms.

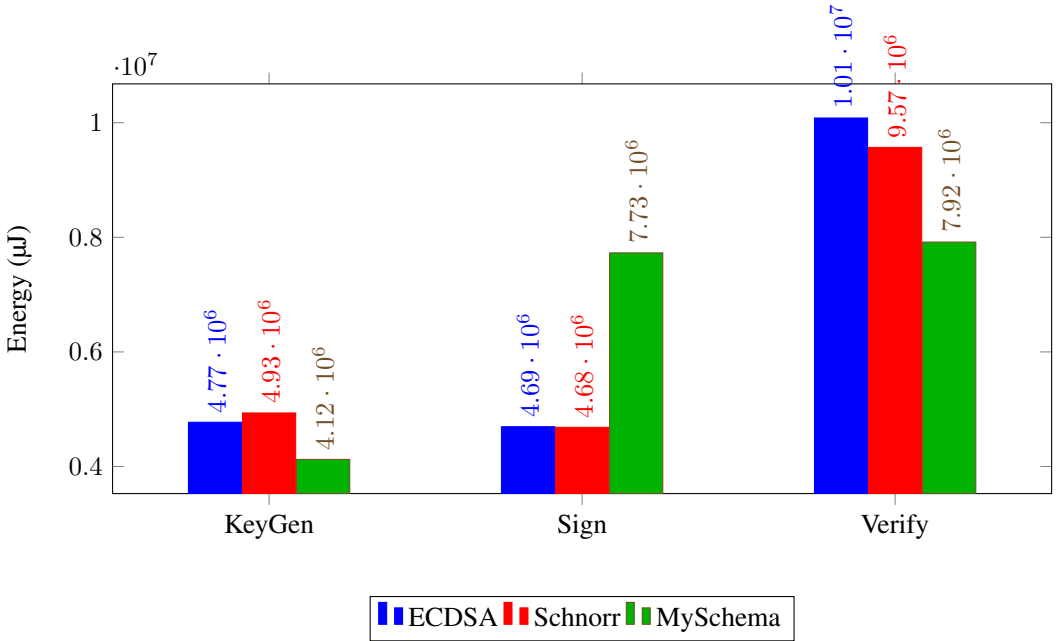


Figure 5. Comparison of approximate energy consumption for each scheme on a simulated Raspberry Pi 4.

Despite the higher time and energy cost due to the complexity of the scheme, it offers several advantages for IoT environments: it allows reducing the curve order while maintaining reasonable security, which decreases key and signature sizes. Furthermore, the underlying problem structure makes the scheme more resistant to brute-force

attacks. Thus, it provides configurable security according to device constraints and is fully feasible on a simulated Raspberry Pi, demonstrating its suitability for resource-constrained IoT applications.

6. Conclusion and futur work

In this paper, we presented a new classical digital signature scheme based on elliptic curves and inspired by the *Hidden Shift Problem* (HSP) and hidden number problems in elliptic groups; it is not post-quantum secure.

The proposed scheme follows a simple Schnorr-like structure while introducing two secret components ((x,Q)) and auxiliary commitments that make extracting the private key much harder. Key generation, signing, and verification algorithms were fully described, and a correctness proof shows that any honest signature is always accepted by the verifier.

We also provided a formal security analysis: in the *Random Oracle* model, using the *forking lemma*, we showed that any effective EUF-CMA attack would recover the private key with non-negligible probability, which would contradict the hardness of the underlying problem.

In conclusion, this scheme is a promising approach to improve the security of digital signatures based on a problem harder than the classical discrete logarithm, while keeping the efficiency of elliptic curves. It provides a good balance between performance, compactness, and security. Future work includes stronger proofs, optimizations, countermeasures, and dedicated cryptanalysis to make this approach even stronger and enable lightweight cryptographic primitives for constrained environments.

REFERENCES

1. W. Stallings, "Format-preserving encryption: Overview and NIST specification," *Cryptologia*, vol. 41, no. 2, pp. 137–152, Mar. 2017, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01611194.2016.1169457. [Online]. Available: https://doi.org/10.1080/01611194.2016.1169457
2. P. Rana and B. P. Patil, "Cyber security threats in IoT: A review," *Journal of High Speed Networks*, vol. 29, no. 2, pp. 105–120, 2023, _eprint: https://journals.sagepub.com/doi/pdf/10.3233/JHS-222042. [Online]. Available: https://journals.sagepub.com/doi/abs/10.3233/JHS-222042
3. A. Djenna, S. Harous, and D. E. Saidouni, "Internet of Things Meet Internet of Threats: New Concern Cyber Security Issues of Critical Cyber Infrastructure," *Applied Sciences*, vol. 11, no. 10, p. 4580, Jan. 2021, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/11/10/4580
4. A. Zannou, A. Boulaalam, and E. H. Nfaoui, "Data flow optimization in the internet of things," *Statistics, Optimization amp; Information Computing*, vol. 10, no. 1, pp. 93–106, Feb. 2022. [Online]. Available: http://www.iapress.org/index.php/soic/article/view/1166
5. A. Marouan, M. Badrani, A. Zannou, N. Kannouf, and A. Chetouani, "E-voting system based on blockchain for enhanced university elections," *SN Computer Science*, vol. 6, no. 3, 2025, cited by: 7. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85218703284&doi=10.1007%2fs42979-025-03671-5&partnerID=40&md5=e122a186b85cdf9e351b38e5773e8e7f
6. J. Samandari and C. Gritti, "Online/Offline Digital Signatures: A Systematic Literature Review," *IEEE Access*, vol. 13, pp. 90 991–91 011, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/11006084
7. C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, Jan. 1991. [Online]. Available: https://doi.org/10.1007/BF00196725
8. T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1057074
9. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. [Online]. Available: https://dl.acm.org/doi/10.1145/359340.359342
10. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Sep. 2012. [Online]. Available: https://doi.org/10.1007/s13389-012-0027-1
11. D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Aug. 2001. [Online]. Available: https://doi.org/10.1007/s102070100002
12. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.
13. H. Vranken, "Sustainability of bitcoin and blockchains," *Current Opinion in Environmental Sustainability*, vol. 28, pp. 1–9, Oct. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877343517300015
14. Z. Wang, H. Jin, W. Dai, K.-K. R. Choo, and D. Zou, "Ethereum smart contract security research: survey and future research opportunities," *Frontiers of Computer Science*, vol. 15, no. 2, p. 152802, Oct. 2020. [Online]. Available: https://doi.org/10.1007/s11704-020-9284-9
15. S. Noether, A. Mackenzie, and t. M. Research Lab, "Ring Confidential Transactions," *Ledger*, vol. 1, pp. 1–18, Dec. 2016. [Online]. Available: https://www.ledgerjournal.org/ojs/ledger/article/view/34

16. H. Song, Y. Wei, Z. Qu, and W. Wang, "Unveiling Decentralization: A Comprehensive Review of Technologies, Comparison, Challenges in Bitcoin, Ethereum, and Solana Blockchain," in *2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 6, May 2024, pp. 1896–1901, iSSN: 2693-2776. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10575445
17. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 51–68. [Online]. Available: https://dl.acm.org/doi/10.1145/3132747.3132757
18. D. J. Bernstein, "Curve25519: New Diffie-Hellman Speed Records," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Berlin, Heidelberg: Springer, 2006, pp. 207–228.
19. L. Chen, D. Moody, A. Regenscheid, and K. Randall, "Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 800-186 (Withdrawn), Oct. 2019. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/186/ipd
20. M. O. Ozmen, A. A. Yavuz, and R. Behnia, "Energy-aware digital signatures for embedded medical devices," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 55–63.
21. S. E. Nouma and A. A. Yavuz, "Post-quantum forward-secure signatures with hardware-support for internet of things," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 4540–4545.
22. A. A. Yavuz, S. Darzi, and S. E. Nouma, "Liteqsign: Lightweight and quantum-safe signatures for heterogeneous iot applications," *IEEE Access*, vol. 13, pp. 171 442–171 456, 2025.
23. M. Randolph and W. Diehl, "Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman," *Cryptography*, vol. 4, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2410-387X/4/2/15
24. P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in *Advances in Cryptology — CRYPTO '96*, N. Koblitz, Ed. Berlin, Heidelberg: Springer, 1996, pp. 104–113.
25. P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Review*, vol. 41, no. 2, pp. 303–332, 1999, _eprint: https://doi.org/10.1137/S0036144598347011. [Online]. Available: https://doi.org/10.1137/S0036144598347011
26. N. K. Sinai and H. P. In, "Performance evaluation of a quantum-resistant Blockchain: a comparative study with Secp256k1 and Schnorr," *Quantum Information Processing*, vol. 23, no. 3, p. 99, Mar. 2024. [Online]. Available: https://doi.org/10.1007/s11128-024-04272-6
27. O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 34:1–34:40, Sep. 2009. [Online]. Available: https://doi.org/10.1145/1568318.1568324
28. D. Micciancio and O. Regev, "Worst-Case to Average-Case Reductions Based on Gaussian Measures," *SIAM Journal on Computing*, vol. 37, no. 1, pp. 267–302, 2007, _eprint: https://doi.org/10.1137/S0097539705447360. [Online]. Available: https://doi.org/10.1137/S0097539705447360
29. H. Bandara, Y. Herath, T. Weerasundara, and J. Alawatugoda, "On Advances of Lattice-Based Cryptographic Schemes and Their Implementations," *Cryptography*, vol. 6, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2410-387X/6/4/56
30. C. Lomont, "The Hidden Subgroup Problem - Review and Open Problems," Nov. 2004, arXiv:quant-ph/0411037. [Online]. Available: http://arxiv.org/abs/quant-ph/0411037
31. I. Chen and D. Sun, "The dihedral hidden subgroup problem," *Journal of Mathematical Cryptology*, vol. 18, no. 1, p. 20220029, 2024. [Online]. Available: https://doi.org/10.1515/jmc-2022-0029
32. G. Kuperberg, "A Subexponential-Time Quantum Algorithm for the Dihedral Hidden Subgroup Problem," *SIAM Journal on Computing*, vol. 35, no. 1, pp. 170–188, 2005, _eprint: https://doi.org/10.1137/S0097539703436345. [Online]. Available: https://doi.org/10.1137/S0097539703436345
33. O. Regev, "A Subexponential Time Algorithm for the Dihedral Hidden Subgroup Problem with Polynomial Space," Jun. 2004, arXiv:quant-ph/0406151. [Online]. Available: http://arxiv.org/abs/quant-ph/0406151
34. D. Micciancio, "Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 2002, pp. 356–365.
35. J. Meers and J. Nowakowski, "Solving the hidden number problem for CSIDH and CSURF via automated coppersmith," in *Advances in Cryptology – ASIACRYPT 2023*, ser. Lecture Notes in Computer Science, vol. 14441. Singapore: Springer, 2023, pp. 23–52.
36. A. M. Childs and W. v. Dam, "Quantum algorithm for a generalized hidden shift problem," Jul. 2005, arXiv:quant-ph/0507190. [Online]. Available: http://arxiv.org/abs/quant-ph/0507190
37. A. Kerimbayeva, M. Iavich, Y. Begimbayeva, S. Gnatyuk, S. Tynymbayev, Z. Temirbekova, and O. Ussatova, "A lightweight variant of falcon for efficient post-quantum digital signature," *Information*, vol. 16, no. 7, 2025. [Online]. Available: https://www.mdpi.com/2078-2489/16/7/564
38. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, Feb. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839
39. "National Institute of Standards and Technology." [Online]. Available: https://www.nist.gov/
40. D. Soni, K. Basu, M. Nabeel, N. Aaraj, M. Manzano, and R. Karri, "SPHINCS+," in *Hardware Architectures for Post-Quantum Digital Signature Schemes*, D. Soni, K. Basu, M. Nabeel, N. Aaraj, M. Manzano, and R. Karri, Eds. Cham: Springer International Publishing, 2021, pp. 141–162. [Online]. Available: https://doi.org/10.1007/978-3-030-57682-0_9
41. H. Singh, "Code based Cryptography: Classic McEliece," May 2020, arXiv:1907.12754 [cs]. [Online]. Available: http://arxiv.org/abs/1907.12754
42. J. Ding and D. Schmidt, "Rainbow, a New Multivariable Polynomial Signature Scheme," in *Applied Cryptography and Network Security*, J. Ioannidis, A. Keromytis, and M. Yung, Eds. Berlin, Heidelberg: Springer, 2005, pp. 164–175.
43. M. J. Josodipuro, K. V. I. Saputra, and S. Lukas, "Statistical analysis of pollard's rho attack on elliptic curve cryptography," in *2022 1st International Conference on Technology Innovation and Its Applications (ICTIIA)*, 2022, pp. 1–6.