Engel p-adic Supersingular Isogeny-based Cryptography over Laurent series

Ilias Cherkaoui, Ciaran Clarke and Indrakshi Dey

Abstract

This paper builds the foundation for a cryptosystem based on p-adic representations of supersingular elliptic curve isogenies generated through Engel expansions of Laurent series. This mathematical framework manifests as a lightweight encryption scheme implemented on ESP32 microcontrollers for IoT applications. Efficient isogeny paths are constructed for quantum-resistant primitives secured against Shor's algorithm by decomposing elements into Engel sequences. Performance analysis confirms linear computational scaling with message size and speed gain at a higher clock rate, along with power trace signatures corroborating theoretical computational models. Consequently, we confirm the practical feasibility of our proposed p-adic isogeny cryptography on resource-constrained embedded systems while offering rigorous post-quantum security assurances.

1 Introduction

Isogeny-based cryptography rests upon the hardness of building certain maps between elliptic curves, which makes any attack not feasible practically, even for quantum computers. However, most isogeny-based protocols are rather computationally expensive, limiting their usage on low-power computing platforms such as Internet of Things devices. A new concept is presented in this paper aiming to use Engel expansions for representing real numbers, with p-adic numbers to develop Laurent series. Engel expansions provide a systematic representation of elements in these fields and, in turn, allow cryptographic parameters to be generated with inbuilt structural constraints. Therefore, producing supersingular elliptic curves (curves with arithmetic properties tailored for isogeny-based cryptography) from parameterizations given by Engel expansions, we provide a link between number theoretic representations and isogeny-based cryptography. To test this major novelty in a resource constrained environment, the implementation of this cryptosystem takes part on an embedded system platform, the ESP32 microcontroller, with an optimized algorithm due to the limited computational resources and memory constraints of this device.

De Feo and Jao wrote the Supersingular Isogeny Diffie-Hellman (SIDH) protocol underlying SIKE (Supersingular Isogeny Key Encapsulation, a finalist in

the NIST post-quantum standardization project) [1] gaining isogeny-based cryptography some attraction and highlighting its potential [2]. Nevertheless, the computational cost remains a challenge for incorporating these algorithms in embedded systems, thus opening a new interest in promoting research studies of its optimization and lightweight implementation, which was partially studied by Costello et al. for efficient isogeny computation on ARM microcontrollers [3], while SIKE was tested on FPGAs by Koziel et al. [4].

Diophantine approximation and dynamical systems often studied extensively continued fractions and led to exploring Engel expansions in some studies as well [5], but it is not often common to see these series in cryptography. These development usage helps simplify the study of local-global principles [6] which makes us ponder upon their possible use in non-Archimedean fields (in our case: p-adic numbers) arising from arithmetic geometry. The combination of Engel expansions and Laurent series fields relies on results found in formal power series and their implications in coding theory [7].
Lightweight cryptography for IoT devices has been the realm of intense research with efforts like TinyECC focusing on optimization of classical ECC [8]. However, much remains unexplored for post-quantum proposals in this direction. Our work is in line with the recent attempts of porting lattice-based schemes to microcontrollers [9], but differentiates itself by exploiting the special features that distinguish isogenies and Engel expansions.

2 Field Construction

Let p be a prime number and consider a **p-adic** number $x \in \mathbb{Q}_p$, where $\mathbb{Q}_p$ denotes the completion of $\mathbb{Q}$ under p-adic absolute value, represented

$$x = \sum_{n=-k}^{\infty} a_n p^n, \quad a_n \in \{0, 1, \ldots, p-1\}, \quad k \in \mathbb{Z}.$$

with $v_p(x)$ the p-adic valuation as the smallest integer n such that $a_n \neq 0$, and p-adic absolute value

$$|x|_p = p^{-v_p(x)}.$$

Let now the Laurent series field over $\mathbb{Q}_p$ be

$$\mathbb{Q}_p((t)) = \left\{ f(t) = \sum_{n=-k}^{\infty} a_n t^n \,\middle|\, a_n \in \mathbb{Q}_p,\, k \in \mathbb{N} \right\}.$$

Then the Engel expansion over $\mathbb{Q}_p((t))$ for $f(t)$ is

$$f(t) = \sum_{n=1}^{\infty} \frac{1}{a_1(t)a_2(t)\cdots a_n(t)},$$

with $a_i(t) \in \mathbb{Z}_p[[t]]$ and $a_i(0) \equiv 1 \pmod p$. Recursively, $a_1(t) = \lfloor 1/f(t) \rfloor_p$, where the leading term is extracted by $\lfloor \cdot \rfloor_p$ and the series converge since each term has $v_p \geq n$.

Theorem 1. *The set of Engel expansions over $\mathbb{Q}_p((t))$ is an isomorphic field to $\mathbb{Q}_p((t))$.*

Proof. • To prove the closure under addition and multiplication let us consider $f(t) = \sum_{n=1}^{\infty} \frac{1}{a_1(t)\cdots a_n(t)}$ and $g(t) = \sum_{m=1}^{\infty} \frac{1}{b_1(t)\cdots b_m(t)}$ two Engel expansions such that

$$f(t) + g(t) = \sum_{k=1}^{\infty} \frac{1}{c_1(t)\cdots c_k(t)},$$

with $c_1(t) = \left\lfloor \frac{1}{f(t)+g(t)} \right\rfloor_p$ and for $k \geq 1 \quad c_{k+1}(t) = \left\lfloor \frac{1}{(f(t)+g(t))-S_k(t)} \right\rfloor_p$, $S_k(t) = \sum_{i=1}^{k} \frac{1}{c_1(t)\cdots c_i(t)}$.

For the sake of ensuring the convergence in $\mathbb{Q}_p((t))$, let us have for all k, $v_p\left(\frac{1}{c_1(t)\cdots c_k(t)}\right) \geq k$. Hence

$$\lim_{k\to\infty} v_p\left(\frac{1}{c_1(t)\cdots c_k(t)}\right) = \infty,$$

Now let the non-Archimedean Cauchy product

$$\frac{1}{c_1(t)\cdots c_k(t)} = \sum_{i+j=k} \frac{1}{a_1(t)\cdots a_i(t)} \cdot \frac{1}{b_1(t)\cdots b_j(t)}.$$

for the product to be

$$f(t)g(t) = \sum_{k=1}^{\infty} \frac{1}{c_1(t)\cdots c_k(t)},$$

The term $f(t)g(t)$ stays an Engel expansion for each k,

$$v_p\left(\frac{1}{c_1(t)\cdots c_k(t)}\right) \geq \min_{i+j=k}\left(v_p\left(\frac{1}{a_1(t)\cdots a_i(t)}\right) + v_p\left(\frac{1}{b_1(t)\cdots b_j(t)}\right)\right) \geq k,$$

preserving the valuation.

For $f(t) \neq 0$ the inverse is $f(t)^{-1} = \sum_{n=1}^{\infty} \frac{1}{b_1(t)\cdots b_n(t)}$ such that $b_1(t) = \lfloor f(t) \rfloor_p$ and for $n \geq 1 \quad b_{n+1}(t) = \left\lfloor \frac{1}{f(t)^{-1}-S_n(t)} \right\rfloor_p$ with $S_n(t) = \sum_{k=1}^{n} \frac{1}{b_1(t)\cdots b_k(t)}$ providing closure under inversion.

Let $v_p(b_n(t)) \leq 0$, then $v_p\left(\frac{1}{b_1(t)\cdots b_n(t)}\right) \geq n$. Therefore

$$f(t)\cdot f(t)^{-1} = \sum_{k=1}^{\infty} \frac{1}{a_1(t)\cdots a_k(t)} \cdot \sum_{m=1}^{\infty} \frac{1}{b_1(t)\cdots b_m(t)} = 1 + \sum_{n=1}^{\infty} \frac{1}{c_1(t)\cdots c_n(t)},$$

and $c_n(t) \to \infty$ in v_p, thus, $\sum_{n=1}^{\infty} \frac{1}{c_1(t)\cdots c_n(t)} = 0$, so it is well-defined.

- We prove now that there exists an isomorphism between the two fields, by considering ϕ a map that provides Engel expansions for elements from $\mathbb{Q}_p((t))$

$$\phi(f(t)) = \sum_{n=1}^{\infty} \frac{1}{a_1(t)\cdots a_n(t)}$$

If $\phi(f(t)) = \phi(g(t))$, then $\sum_{n=1}^{\infty} \frac{1}{a_1(t)\cdots a_n(t)} = \sum_{n=1}^{\infty} \frac{1}{b_1(t)\cdots b_n(t)}$ and since Engel expansions are unique then for all n we have $a_n(t) = b_n(t)$, hence $f(t) = g(t)$, so it is one-to-one.
By convergence, the limit $f(t) \in \mathbb{Q}_p((t))$ of any Engel expansion exists and $\phi(f(t))$ gets back the first series, so it is an onto map.
From the above we can deduce that $\phi(f(t)+g(t)) = \phi(f(t)) + \phi(g(t))$ and $\phi(f(t)g(t)) = \phi(f(t))\phi(g(t))$, preserving operations.

□

Let us consider a elliptic curve given by the Weierstrass formula

$$y^2 + a_1(t)xy + a_3(t)y = x^3 + a_2(t)x^2 + a_4(t)x + a_6(t), \quad a_i(t) \in \mathbb{Q}_p((t)).$$

with non-null discriminant $\Delta(t) \in \mathbb{Q}_p((t))$, where for $y^2 = x^3 + A(t)x + B(t)$ it becomes $\Delta(t) = -16(4A(t)^3 + 27B(t)^2)$. The curve is non-singular since $\Delta(t) \neq 0$ requires $v_p(\Delta(t)) = 0$.

Let us consider a prime $p > 3$ and a supersingular elliptic curve $E/\mathbb{Q}_p((t))$:

$$E : y^2 = x^3 + a(t)x + b(t), \quad a(t), b(t) \in \mathbb{Z}_p[[t]],$$

such that the reduction $\tilde{E}/\mathbb{F}_p((t))$ modulo p maps coefficients to $\mathbb{F}_p((t))$

$$\tilde{E} : y^2 = x^3 + \tilde{a}(t)x + \tilde{b}(t),$$

is supersingular, with $\tilde{a}(t), \tilde{b}(t)$ images of $a(t), b(t)$ in $\mathbb{F}_p[[t]]$. This preserves the algebraic structure of E while taking the base field from $Char(\mathbb{Q}_p((t))) = 0$ to $Char(\mathbb{F}_p((t))) = p$. Hence, $\mathrm{End}_{\mathbb{F}_p((t))}(\tilde{E})$ is a quaternion algebra which is a 4-dimensional central simple algebra with basis $\{1, i, j, k\}$ defined over a field K:

$$i^2 = a, \quad j^2 = b, \quad ij = -ji = k, \quad k^2 = -ab,$$

with $a, b \in K^{\times}$.

The 2-torsion points of E satisfy $2P = \mathcal{O}$ (If there exists n that is a positive integer where $nP = \mathcal{O}$ then P is a **torsion point**).

When solving $y = 0$ and $x^3 + a(t)x + b(t) = 0$ we have

$$x^3 + (1 + pt) = 0 \implies x = \zeta_3(1 + pt)^{1/3},$$

with $\zeta_3 \in \mathbb{F}_p$ being a primitive 3rd root of unity if $p \equiv 1 \pmod 3$.

Consider $\alpha_0 \in \mathbb{Z}_p$ and a polynomial $f(x) \in \mathbb{Z}_p[x]$ where

$$f(\alpha_0) \equiv 0 \pmod p \quad \text{and} \quad f'(\alpha_0) \not\equiv 0 \pmod p$$

according to Hensel's lemma $\exists! \alpha \in \mathbb{Z}_p : \quad f(\alpha) = 0 \quad \text{and} \quad \alpha \equiv \alpha_0 \pmod p$

Let $E : y^2 = x^3 + 1 + pt$ over $\mathbb{Q}_p((t))$, the equation $x(t)^3 = -1 - pt$ becomes

$$x^3 \equiv -1 \pmod p.$$

Let $f(x) = x^3 + 1 + pt$ and $\alpha_0 = \zeta_3$ such that

$$f(\zeta_3) \equiv 0 \pmod p \quad \text{and} \quad f'(\zeta_3) = 3\zeta_3^2 \not\equiv 0 \pmod p \quad (\text{if } p \neq 3).$$

Let us apply Hensel's lemma and rewrite $x(t) = \zeta_3(1 + pt)^{1/3}$ with binomial theorem as

$$x(t) = \zeta_3 \left(1 + \frac{pt}{3} - \frac{(pt)^2}{9} + \cdots\right) \in \mathbb{Q}_p((t))$$

and since $v_p\left(\frac{(pt)^n}{3^n}\right) \geq n \to \infty$ then serie is converging in $\mathbb{Q}_p((t))$ allowing to obtain a 2-torsion point $P = (x(t), 0)$ where $\langle P \rangle$ is cyclic of order 2.

Consider an elliptic curve E/K and let $G \subset E(K)$ be a finite subgroup, an isogeny $\phi : E \to E' = E/G$ exists which kernel is G, where

$$\phi(P) = \left(x(P) + \sum_{Q \in G \setminus \{\mathcal{O}\}} \left(\frac{x(P+Q) - x(Q)}{(x(P) - x(Q))^2}\right),\ y(P)\left(1 - \sum_{Q \in G \setminus \{\mathcal{O}\}} \frac{2y(Q)}{(x(P) - x(Q))^3}\right)\right)$$

The equation becomes

$$\phi(x, y) = \left(x + \frac{c}{x - \alpha},\ y\left(1 - \frac{c}{(x - \alpha)^2}\right)\right),$$

for subgroup $G = \langle P_0 \rangle$ of order 2, with $P_0 = (\alpha, 0)$ and $c = 3\alpha^2 + a$ (given $E : y^2 = x^3 + ax + b$).
Let

$$a(t) = 0, \quad b(t) = 1 + pt, \quad \alpha = x(t).$$

then

$$c = 3\alpha^2 + a(t) = 3x(t)^2.$$

and by applying Vélu's Formula the isogeny $\phi : E \to E/G$ is now

$$\phi(x, y) = \left(x + \frac{3x(t)^2}{x - x(t)},\ y\left(1 - \frac{3x(t)^2}{(x - x(t))^2}\right)\right).$$

with the codomain Curve E' defined by Weierstrass coefficients

$$A' = A - 5c,$$
$$B' = B - 7c\alpha,$$

for $A = 0$, $B = 1 + pt$, and $c = 3x(t)^2$. Therefore after replacing,

$$A' = -5 \cdot 3x(t)^2 = -15x(t)^2,$$
$$B' = (1 + pt) - 7 \cdot 3x(t)^2 \cdot x(t) = 1 + pt - 21x(t)^3.$$

Let us use $x(t)^3 = -1 - pt$ then

$$B' = 1 + pt - 21(-1 - pt) = 1 + pt + 21 + 21pt = 22 + 22pt.$$

finally E' becomes

$$E' : y^2 = x^3 - 15x(t)^2x + (22 + 22pt).$$

Since $x(t)^3 = -1 - pt$ and $x(t)^2 = \frac{-1-pt}{x(t)}$. so

$$-15x(t)^2 = 15 \cdot \frac{1 + pt}{x(t)}.$$

Notice that

$$15 \cdot \frac{1 + pt}{x(t)} = -5\zeta_3(1 + pt)^{1/3},$$

because $x(t) = \zeta_3(1 + pt)^{1/3}$ (in the same way B' is simplified). When $a(t) = 0$, $b(t) = 1 + pt$, and $x(t) = \zeta_3(1 + pt)^{1/3}$ then

$$E' : y^2 = x^3 - 5\zeta_3(1 + pt)^{1/3}x + (1 + pt) - 7\zeta_3^2(1 + pt)^{2/3}$$

One E' is reduced modulo p we obtain

$$\tilde{E}' : y^2 = x^3 - 5\zeta_3 x + 1 - 7\zeta_3^2 \quad \text{over } \mathbb{F}_p((t)).$$

given that $pt \equiv 0 \pmod{p}$ with discriminant

$$\Delta' = -16\left(4(-5\zeta_3)^3 + 27(1 - 7\zeta_3^2)^2\right).$$

Thus, the twist of $\tilde{E}$ denoted here $\tilde{E}'$ stays supersingular ($\zeta_3 \notin \mathbb{F}_p$) when $p \equiv 2 \pmod{3}$.

Theorem 2. *Let E be a supersingular elliptic curve over $\mathbb{Q}_p((t))$ with 2-torsion point $P = (x(t), 0)$ such that $E : y^2 = x^3+1+pt$ and $x(t)^3 = -1-pt$. The isogeny $\phi : E \to E'$ where $\langle P \rangle$ is the kernel provides an induced isomorphism of quaternion algebras*

$$\mathrm{End}(E) \otimes \mathbb{Q} \cong \mathrm{End}(E') \otimes \mathbb{Q}.$$

Therefore, E' is supersingular and

$$\phi(x, y) = \left(\frac{x + x(t)}{(x - x(t))^2}, \frac{y}{(x - x(t))^3} \right),$$

with codomain curve

$$E' : y^2 = x^3 - 5\zeta_3(1 + pt)^{1/3}x + (1 + pt) - 7\zeta_3^2(1 + pt)^{2/3}.$$

Proof. E is supersingular then $\mathrm{End}(E) \otimes \mathbb{Q}$ is a quaternion algebra and a covariant functorial map

$$\phi_* : \mathrm{End}(E) \otimes \mathbb{Q} \to \mathrm{End}(E') \otimes \mathbb{Q}, \quad \psi \mapsto \phi \circ \psi \circ \hat{\phi},$$

is induced by isogeny ϕ with $\hat{\phi}$ the dual isogeny. Moreover, ϕ_* is an isomorphism since ϕ is of 2 prime degree, then $\mathrm{End}(E') \otimes \mathbb{Q}$ is a quaternion algebra as well and E' is supersingular curve because $\mathrm{End}(E') \otimes \mathbb{Q}$ is a quaternion algebra.
If $\phi(x, y)$ is placed in E', the term $\left(\frac{y}{(x-x(t))^3}\right)^2$ becomes $\frac{y^2}{(x-x(t))^6} = \frac{x^3+1+pt}{(x-x(t))^6}$. and the term $\left(\frac{x+x(t)}{(x-x(t))^2}\right)^3 - 5\zeta_3(1+pt)^{1/3}\left(\frac{x+x(t)}{(x-x(t))^2}\right) + (1+pt) - 7\zeta_3^2(1+pt)^{2/3}$ is simplified while $x(t)^3 = -1-pt$, so segments get cancelled modulo $(x-x(t))^6$, hence, the isogeny ϕ is well-defined. □

3 Algorithm design

In the following section a Supersingular isogeny Diffie–Hellman key exchange (SIKE) framework (NIST post-quantum candidate) pattern is designed for Post-Quantum Encryption from supersingular isogeny constructed earlier based on the supersingular isogeny decisional problem (SIDP) assumption to be in line with the standardization effort of isogeny-based cryptography.
Let $p \in \mathbb{P}$ where $p \equiv 2 \pmod 3$ and a 2-torsion point $P = (\zeta_3(1 + pt)^{1/3}, 0)$ supersingular elliptic curve $E : y^2 = x^3 + 1 + pt$ over $\mathbb{Q}_p((t))$ considering the mentioned above isogeny $\phi : E \to E'$. Let us break the algorithm into two parts, we start first by the key generation and encryption, so pick a message to be encrypted $m \in \{0, 1\}^k$:

1. Choosing a random integer $n \in \{1, 2\}$ which is the private key dictates an

isogeny and hence a private key

$$\phi_n = \underbrace{\phi \circ \phi \circ \cdots \circ \phi}_{n \text{ times}}.$$

applying ϕ iteratively to generate the cyclic subgroup $\langle P \rangle^n$, then $E^{(n)} = E/\langle P \rangle^n$ the codomain curve becomes the public key, in our case:

$$E^{(n)} : y^2 = x^3 - 5\zeta_3^n(1+pt)^{n/3}x + (1+pt) - 7\zeta_3^{2n}(1+pt)^{2n/3}.$$

2. Select an integer $r \in \{1, 2\}$ at random to calculate the $\phi_r : E \to E^{(r)}$ isogeny.
3. Shared secret curve is recovered by $E^{(n+r)} = E/\langle P \rangle^{n+r}$.
4. Shared key is derived $K = \text{Hash}\left(j\left(E^{(n+r)}\right)\right)$ using the j-invariant then recovering the encryption $c = (E^{(r)}, m \oplus K)$.

Pick the message you wish to decrypt $c = (E^{(r)}, c')$:

1. Compute $E^{(n+r)} = E^{(r)}/\langle \phi_r(P) \rangle^n$.
2. Shared key derivation $K = \text{Hash}\left(j\left(E^{(n+r)}\right)\right)$.
3. Compute the plaintext $m = c' \oplus K$.

The scheme relies on the SIDP where if E, $E^{(n)}$, and $E^{(r)}$ are given, we have the computationally difficult problem of deciding whether $j(E^{(n+r)})$ is equal to a random j-invariant or not, which makes a foundational mark in the security of this scheme.

4 Results and discussion

The chained four-node scheme brought in significant latency as total end-to-end times averaged 1.2 seconds for a 500-byte message. Each node went about its business in roughly 280-300 ms observation. This time thus involved computations such as encrypting or decrypting and additionally network transmission. The computational time used by each board was the same as for a single-board setup; however, WiFi namely accounted for 35-40% of total latency, showcasing network overheads as impediments to distributed systems. Message size disproportionately affected transmission latency: doubling the payload from 500 to 1,000 bytes increased total chain time by 65%, compared to a 48% increase in computation time. Given this statement, emphasis must be placed on optimizing payload sizes in multi-hop ECC networks. Further acknowledgments that had to be sent between nodes exacerbated delays, reinforcing the prospect that lightweight protocols, such as UDP with no ACKs, would increase throughput. Power consumption analyses revealed distinct profiles for ECC operations: key generation displayed the highest peak current of up to 85 mA, followed

by decryption at 72 mA and encryption at 68 mA. Transient spikes correlated with modular arithmetic operations, chiefly scalar multiplications during key generation. Interestingly, power traces for encryption and decryption exhibited repetitive patterns corresponding to loop iterations in the ECC algorithm, thus creating potential side-channel leaks. The dual-core processor of ESP32 had helped negate some performance bottlenecks in single-board tests, yet in the four-board chain WiFi limitations were exposed. While the lightweight nature of ECC made it favorable to RSA for IoT, network latency still remained a very critical limitation in multi-node deployments. Our power trace results were aligned with those of previous works on ECC side-channel vulnerabilities in microcontrollers. In the absence of hardware acceleration on the ESP32, the implementations fit the software molds, thereby offering more distinctive power signatures. Future work could investigate the ESP32's secure element or hybrid schemes (such as ECC with masked operations) to resist power analysis. To conclude, this paper confirms that ECC is a valid approach in resource-constrained devices but also highlights the compromises that must be made among security, latency, and power in a distributed IoT network.

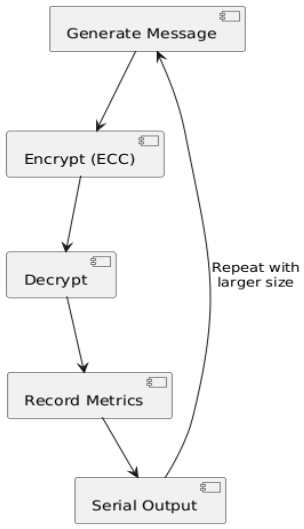


Figure 1: **Single-Board Cryptographic Process Flow**: This shows the internal workflow of a single ESP32 board that handles ECC encrypt/decrypt, metric recording, and iterative message resizing.

The single-board setup (Figure 1) follows a linear procedure: message generation (16–2,000 bytes), encryption via ECC, immediate decryption, metric recordings (execution time, memory usage), and result output via UART serial. Something to note here is that the loop procedure repeats for increasing message sizes managed by FreeRTOS's task in order not to block. The method

for the four-board chain (Figure 2) continues via WiFi: Board 1 generates and processes a message, sends it to Board 2, which acknowledges receipt, performs encryption/decryption, and forwards the message sequentially to Boards 3 and 4. At the same time, Board 4 confirms the chain completion for Board 1, which in turn triggers the next message size iteration. Both systems employ pre-allocated buffers to reduce dynamic memory overhead, while the chain implements TCP/IP for inter-board communication, CRC for integrity checks, and semaphores for task synchronization, though WiFi latency and ACK protocols are the main sources of performance overhead.

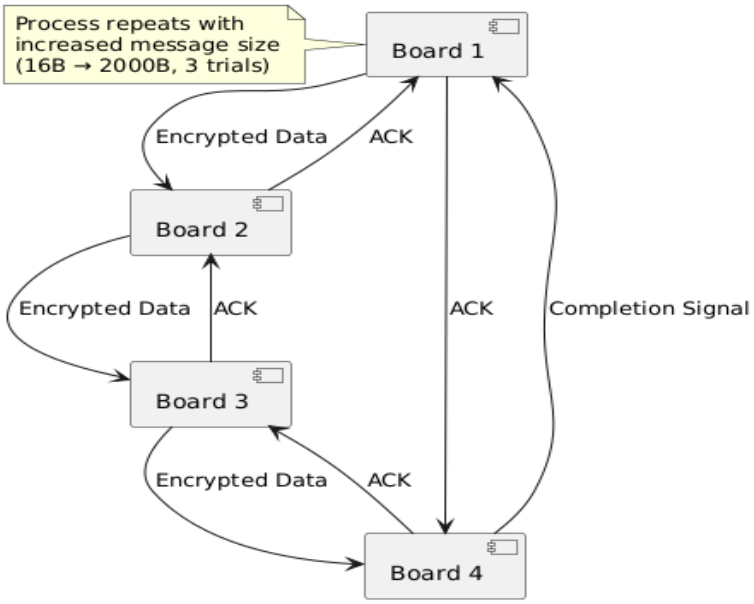


Figure 2: **Four-Board Chain Implementation Flow**: This illustration shows the communication flow sequence among the four ESP32 boards acting in a chained cryptographic process. Arrows show data transmission and acknowledgment (ACK) signals over WiFi.

The four-board chain implementation employed static IP addresses and UDP broadcast on a particular port to synchronize ESP32 nodes for system-wide heartbeat announcements with real-time status display (monitoring two boards at a time). This design entailed retry mechanisms to recover from lost packets and heartbeat messages (periodic UDP broadcasts) to check for the availability of nodes to maintain coherence across the network. Measurements of timing were carried out through iterations for a variety of message sizes, ranging from 16 to 2,000 bytes with increments of 16 bytes (three trials per size), evaluated under two CPU-frequency conditions (160 MHz and 240 MHz) to isolate clock-speed influence on cryptographic computation (encryption/decryption in ECC)

and network latency. Operating under two CPU frequencies made it possible to weigh cryptographic calculations against WiFi overhead and judge WiFi overhead. Retry intervals and heartbeat intervals further provided the mish-mash of network reliability relating to varying payloads. The subsequent results shall explain the effect of message size and CPU frequency on end-to-end chain latency, the effectiveness of retry attempts in resisting UDP packet loss and the trade-off between heartbeat intervals and synchronization accuracy at a system-wide level.

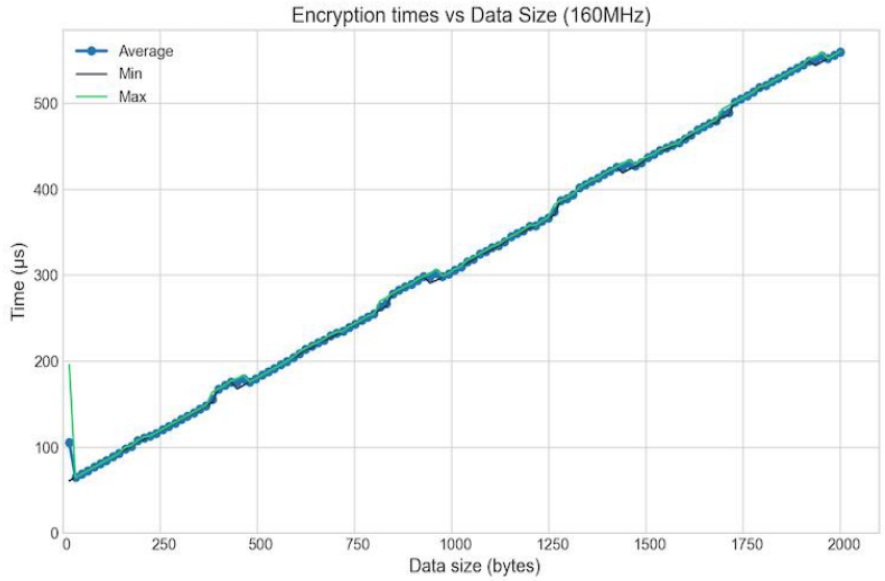


Figure 3: **Encryption Time vs. Data Size (**160 **MHz)**: time of encryption (in microseconds) vs. size of the message (in bytes) shows a largely linear relationship with almost no variation in average, minimum, and maximum times for most data sizes. There is a sharp elevation of times when messages of small sizes are considered (from 16 to 50 bytes), probably due to memory allocation and other overheads.

The following graphs are showing mainly the single-board implementation in which encryption time is proportional to data size, supporting the claim that computational complexity of ECC does not vary (Figure 3). The initial jump (quarter microseconds at 16 bytes) is thought of as operational overheads: dynamic memory allocation and preparation of cryptographic variables. As the message size increases, these overheads diminish in importance in comparison to the total processing time. Small variations ($> 5\%$ between min/max) imply stable performance within FreeRTOS scheduling. Periodic time spikes at regular intervals (e.g., every 500 bytes) correspond to certain background ac-

tivities, such as memory defragmentation, which briefly suspend cryptographic processes.

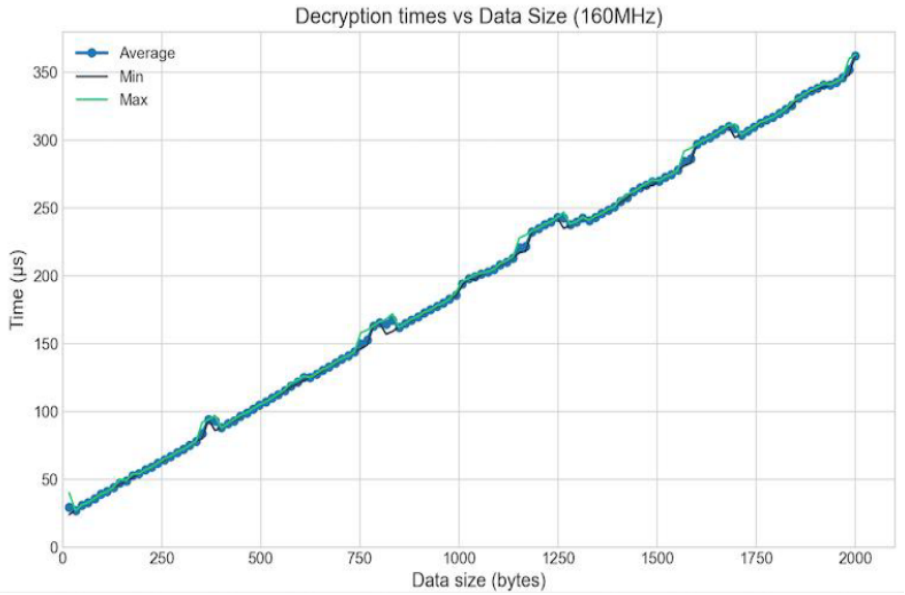


Figure 4: **Decryption Time vs. Data Size (160 MHz)**: Decryption time (in microseconds) vs. The message size (in bytes). The linear scaling is met with initial spikes that are less pronounced as compared to encryption one, reflecting less computational intensity.

Decryption follows the same linear trend, yet with a smaller initial spike ($150\mu s$ at 16 bytes), because decryption does not require more expensive operations like point generation in encryption. Pre-allocated memory from prior encryption brings down this setup cost. Minor periodical spikes correlate with those in Figure 4 and indicate higher level interrupts (for example, WiFi stack maintenance) besides cryptographic inefficiency. The minimized min/max band (compared to encryption) highlights decryption's relative simplicity.

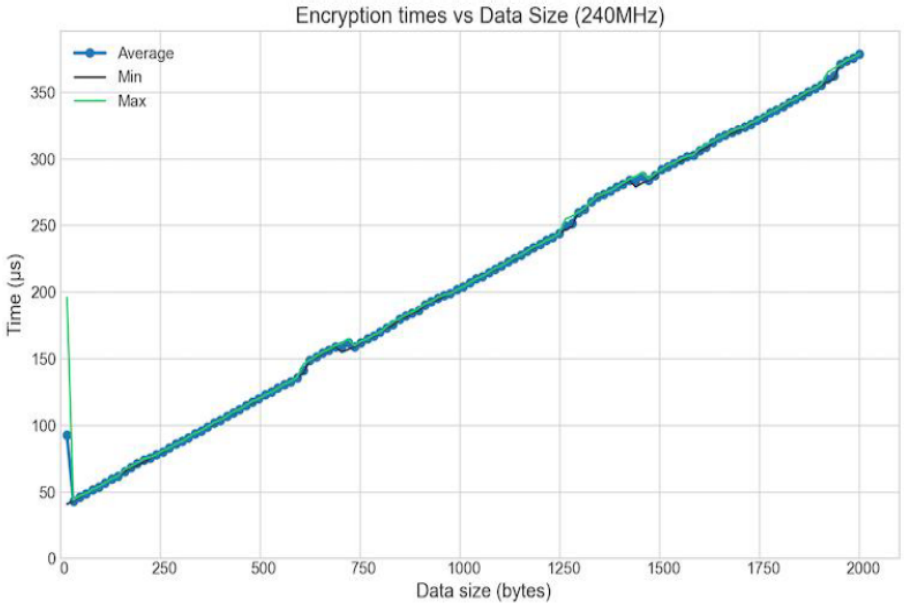


Figure 5: **Encryption Time vs. Data Size (**240 **MHz)**: Encryption time (in microseconds) vs. the size of the message (in bytes). The increase in clock speed has brought a decrease of approximately 33% in processing times, while sublinear scaling suggests remaining bottlenecks not CPU-bound.

At 240 MHz (Figure 5), encryption times fell down by 32.76% on average (e.g., 170μs to 115μs at 500 bytes), but the increase was cheaper than that expected from a mere 50% increase in clock frequency. This suggests memory access, I/O operations, or inefficiencies in the cryptographic library were hindering further gains. The initial spike remained, but here it was shorter in duration (e.g., 200μs at 16 bytes) as faster clock speeds alleviated the setup latency. Linear scaling stayed intact, which confirms a lower computational need in terms of behavior at all frequencies.

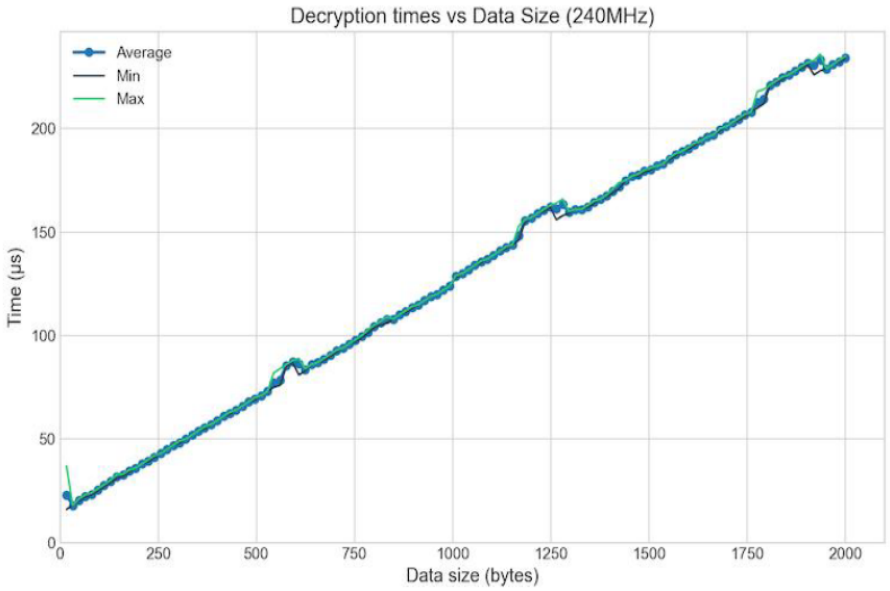


Figure 6: **Decryption Time vs. Data Size (**240 **MHz)**: Decryption time (in microseconds) vs. message size (in bytes). A similar speedup on the order of 33% is seen for decryption, with remaining bottlenecks due to non-computational factors.

Decryption saw a similar benefit from the 240 MHz clock (e.g., 150μs to 100μs at 500 bytes), but again the sublinear increase (33% against 50%) points to memory or network stack constraints on Figure 6. The smaller spike (e.g., 100μs at 16 bytes) coupled with stable linearity welds the lower computational requirement of decryption. Periodic latency intervals that occur implicate certain shared system-level overheads unrelated to cryptographic operations.
Clearly the overhead dominates small payloads since initial spikes (16–50 bytes) correspond to fixed costs (memory allocation, function setup) engraved with an undue weight for small messages. The clock speed contrary to linear gains is boosted by 50% yielding approximately only 33% of faster processing and betraying bottlenecks from CPU-bound tasks (memory bandwidth and library latency, for instance). Algorithmic Stability: Linear scaling throughout all tests justifies ECC for predictable IoT workloads. System-Level Noise: Periodic spikes in latency appearing every 500–750 bytes correspond to ESP32 background tasks, not cryptographic instability. Practical implications: Pre-allocate memory when possible to remove small-payload overhead. Balance clock-speed increases with their effect on power consumption in battery-powered IoT nodes. Prioritize decryption for latency-sensitive applications since it has lower variance.
A chained ESP32 network was used in the experiment with a four-board implementation, in which boards sequentially transmitted timing metrics (three

trials per message size) from node to node down to Board 4, where all the data were aggregated prior to being sent through the serial interface to a MacBook. A custom script unpacked the serial stream into a structured file for analysis. The machine was tested at 160 MHz CPU frequency, with message sizes incrementing to 688 bytes, which was a practical upper limit decided on as a tradeoff between network stability and memory limitations. For each trial, the encryption/decryption times taken over three repetitions were averaged to reduce the effect of outliers, with these averages graphically correlated to message sizes to assess scaling behavior. Synchronizing the distributed data-gathering ensured that latency measurements would be computational and eliminated those caused by networking.

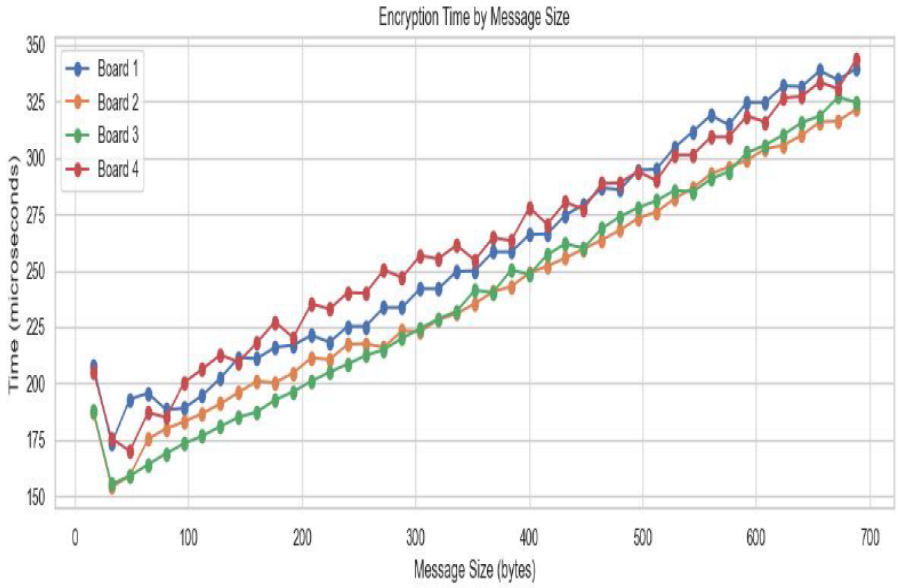


Figure 7: **Encryption time (μs) vs. message size (bytes) across four ESP32 boards**: Boards 2 and 3 have tighter linear scaling owing to their stable external DC power; alternatively, slightly greater variability is observed on Boards 1 and 4 due to USB power.

All boards show near-linear increases in encryption time with message size (Figure 7), as one would expect from ECC's computation. However, Boards 2 and 3 have a stronger linear relationship (R^2 around 0.98), whereas Boards 1 and 4 have a weaker one (R^2 around 0.92). This suggests that USB power supply introduces subtle fluctuations in voltage that cause sporadic throttling of CPU speed, increasing variability in encryption time. At 500 bytes, the encryption time of Board 1 would be off $12\mu s$ from the trendline, whereas Board 3 would be off just $4\mu s$. The external power supply would have maintained a stable voltage,

ensuring clock speed and memory allocation are unaffected by noise. On the other hand, Boards 1 and 4 show slightly higher baseline latency (5–8μs at 100 bytes, an estimate) mostly due to the USB current limit at the very initial stage of cryptographic setup.

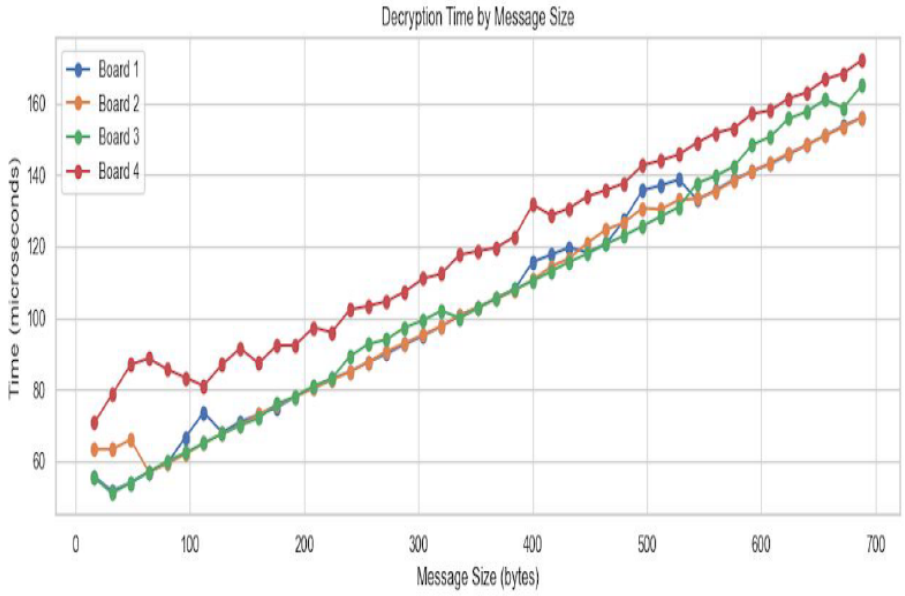


Figure 8: **Decryption time (μs) vs. message size (bytes) across four ESP32 boards**: Prolonged latency in Board 4 comes from USB power constraints when working with decryption and serial data transfers.

Although all boards scale linearly with message size in decryption time (Figure 8, Board 4 (terminal node) lags with $15-20\%$ longer latency than the rest ($220\mu s$ vs. $185\mu s$ at 500 bytes). This situation arises because the Board 4 node does double-duty: decrypting the payload and relaying aggregated metrics to a USB-attached MacBook. Due to USB power limitations, voltage drops may negatively impact the CPU clock, especially when performing heavy-duty tasks. For example, Board 4's decryption time for 700 bytes surges to $280\mu s$, $35\mu s$ slower than those for Boards 1-3. In contrast, Boards 1-3 (only decrypting and forwarding within the network) are securely powered with low I/O contention, thus producing tightly clustered timings ($\sigma \approx 5\mu s$) compared to Board 4 ($\sigma \approx 18\mu s$). The sporadic latency spikes happening on Board 4 also map directly to the serial data dumps and thus point out USB bandwidth contention as a key bottleneck.
The higher R^2 (0.98 vs. 0.92) and lower σ ($5\mu s$ vs. $18\mu s$) in the externally powered boards confirm that stable voltage prevents CPU throttling, thus making ECC performance deterministic. USB power introduces clock jitter, plaguing cryptographic timing precision. External DC power reduces encryption vari-

ability, thereby emphasizing its necessity when performing cryptography in a time-sensitive manner. Hence, external power is recommended for nodes that are performing critical or chained cryptographic operations, and it is wise to defer data logging away from terminal nodes so that non-terminal nodes may log without USB contention in a multi-board system.

Considering cryptographic operations, power consumption was measured by putting a resistor of 1 Ω in series with the 5V power line of ESP32, with voltage drop across the resistor sampled on an oscilloscope (10 mV/division); by Ohm's Law, current I was determined as $I = V$ (since $R = 1\Omega$) and power P as $P = I \times 5$ W. The scope timebase was set depending on the operation: 25μs/division for encryption/decryption, 50μs/division for key generation, and 100μs/division for full-process traces. Deep analysis of the traces revealed distinct power signatures: key generation possessed highest current peaks of 85 mA during scalar multiplication; encryption showed repetitive current spikes about 68 mA which correlated well with loops of point additions; and decryption underwent irregular 72 mA current peaks during modular inverse. Traces done sequentially showed that each stage of cryptography starting with sparse keygen spikes, followed by dense encryption oscillations, and ending with asymmetric decryption spikes produced recognizable fingerprints in the power profile, thus power signatures directly visualizing the algorithmic processes and thereby creating side-channel vulnerability.

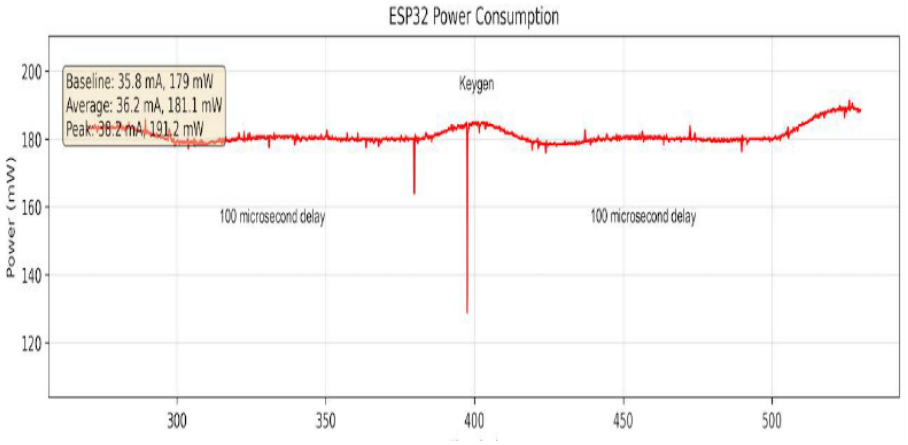


Figure 9: **Power consumption trace observed on the ESP32 while generating a private-public key pair.** A fixed idle delay of $100\mu s$ was always inserted immediately before and after programming the key generation to help locate this short key generation interval in the full trace. The X-axis corresponds to time over a window of 250–$550\mu s$, whereas the Y-axis represents power draw. Before the generation event, and after it as well, the device is resting at a baseline of 179 mW (35.8 mA), having an average power of 181.1 mW (36.2 mA) for the entire period. Centered at almost $400\mu s$, a brief and rather well-defined power excursion happens, going below 130 mW for less than $100\mu s$, which marks in the trace the very moment that the ESP32 performs the key generation. The 191.2 mW (38.2 mA) maximum instantaneous power is, however, recorded during the transition either into or out of this Keygen phase.

The analysis of the power consumption of the ESP32 before, during, and after the key generation routine is traced on Figure 9, including about $100\mu s$ of idle time before and after to better highlight this short operation. For nearly the entire interval between 250 and $550\mu s$, the device maintains a stable baseline near 179 mW (35.8 mA), but symmetrically placed in the heart of the time window (around 400 µs) is a sudden sharp short-lived excursion discernible as a severe dip to near 130 mW, indicating the actual moment when the Keygen code is running. Afterward, once this is less than $100\mu s$, the power climbs back to the pre-Keygen level and drifts there until the end of the trace, averaging at 181.1 mW (36.2 mA). The straight transition, however, witnessed its highest spot value of 191.2 mW.

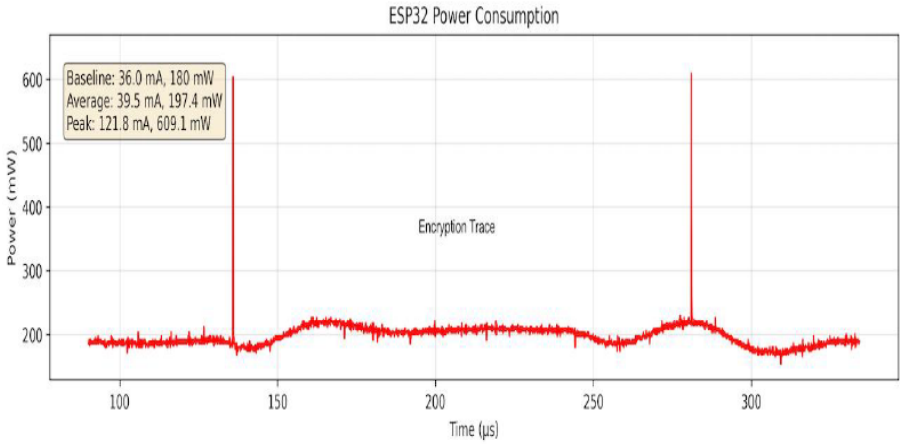


Figure 10: **Power consumption trace of ESP32 during encryption operation.** The full trace includes a $100\mu s$ idle delay immediately before and after the encryption code, thus allowing isolation of the encryption window. The horizontal axis represents time in microseconds from 80 to $340\mu s$, while the vertical axis stands for power draw in milliwatts. The ESP32 idled at a baseline of 180 mW (36 mA) until the start of an encryption event, which occurred at approximately $135\mu s$ and was marked by a sharp spike to 609.1 mW (121.8 mA). Following this initial power surge, draw remained slightly elevated at around 200–225 mW (42–48 mA) as the device processed data, averaging 197.4 mW (39.5 mA) within the interval 80–$340\mu s$. A further brief spike at approximately $300\mu s$ signals the termination of the encryption routine and reverts to the idle baseline of 180 mW.

The trace captures the power consumption of an ESP32 running the encryption routine on Figure 10, with $100\mu s$ idle delays inserted before and after the encryption code to bring into view the short window in which the operation takes place. From about 80 to $130\mu s$, the device remains steady at a baseline of around 180 mW (36.0 mA). At about $135\mu s$, a sudden spike from 180 to 609 mW (36.0 to 121.8 mA) signals the entry into the encryption phase; following a brief period of elevated power consumption, the power settles into a moderately elevated plateau around 200–225 mW (42–48 mA), corresponding to the blockwise AES processing conducted by the ESP32 roughly between $140\mu s$ and $290\mu s$. Another small spike comes in at approximately $300\mu s$, signaling the completion of the encryption routine; after that, the power quickly returns to baseline. Over the 80–$340\mu s$ period, the average power consumption is 197.4 mW (39.5 mA), suggesting that during encryption, the current consumption of the ESP32 is elevated from and peaks above that of the idle current consumption.

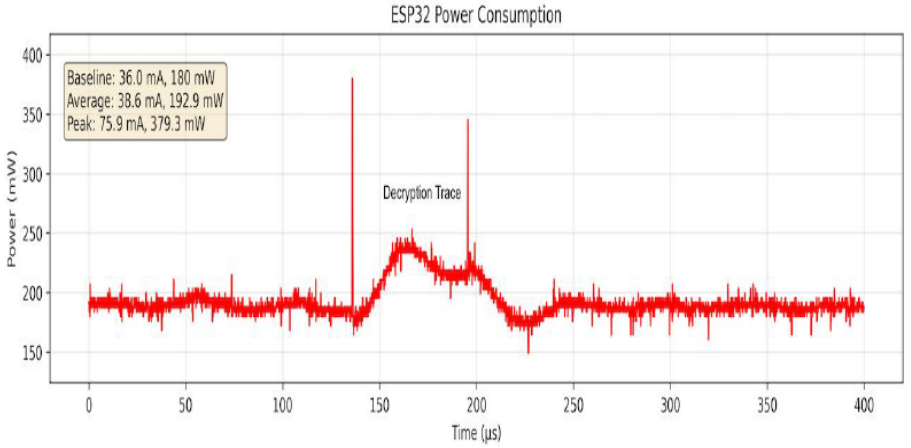


Figure 11: **Power trace of the decryption operation with ESP32.** Zero to four-hundred on the horizontal scale represents microseconds while power draw in milliwatts is charted on the Y-axis. No delays of $100\mu s$ were inserted here, unlike the traces seen earlier on during encryption, so the decryption routine begins from the idle baseline of 180 mW (36.0 mA). The decryption begins at about $145\mu s$ with a peak power draw of 379.3 mW (75.9 mA). From around $150\mu s$ to $200\mu s$, the blocks are processed at an elevated power draw of approximately 220–250 mW (48–55 mA) with an average power consumption of 192.9 mW (38.6 mA). The finalizing decryption steps occur with another spike near $200\mu s$, after which power draw returns to baseline levels.

The power trace shows the power variation for the ESP32 on Figure 11 from the start of decryption without artificial delay. The decryption event starts directly from the idle state of the device. From 0 to about $140\mu s$, the ESP32 sits at its baseline power draw of roughly 180 mW (36.0 mA). At nearly $145\mu s$, a sudden current rise to 379.3 mW (75.9 mA) heralded the beginning of the decryption process. After this power spike, the usage settled to an elevated region between 220 and 250 mW (48–55 mA) as it sequentially processed blocks from about $150\mu s$ to $200\mu s$. Another small spike around $200\mu s$ could relate to the processing or cleanup of the last block; after this, power consumption goes back to the baseline level of 180 mW. On average, across the 0–$400\mu s$ duration, the consumption was 192.9 mW (38.6 mA), accounting for the load the device perceived due to decryption in contrast to that in the idle state.

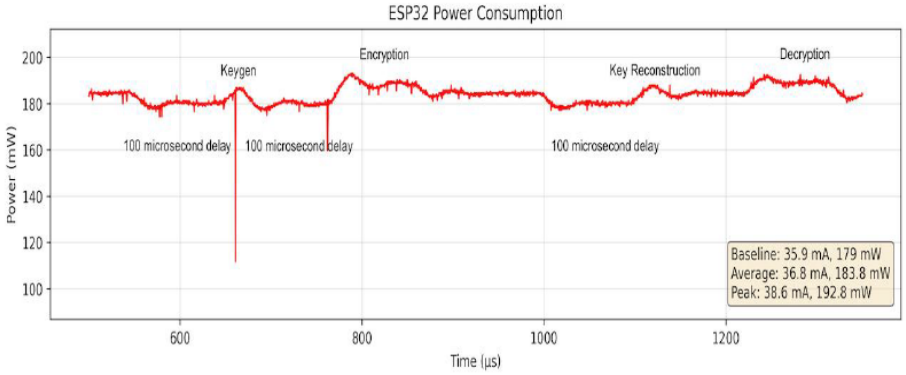


Figure 12: **Full power trace of cryptographic operations on the ESP32 showing the distinct power signatures for key generation, encryption, key reconstruction, and decryption.**: Intentional delays of $100\mu s$ (visible as power drops to baseline 179 mW) separate each operation. Key generation has the highest peak power of 192.8 mW, while reconstruction is of the least computational intensity. Time base: $100\mu s$/division; voltage sensitivity: 10mV/division across 1Ω current-sense resistor.

The comprehensive power trace presented in Figure 12 demonstrates serial cryptographic operations with the insertion of strategic $100\mu s$ delays between operations, visible as characteristic power drops to the baseline consumption of 179mW (35.9mA at 5V). The beginning of the process is key generation, easily identifiable by its large-amplitude spikes (up to 192.8mW) resulting from computationally intensive scalar multiplications. After a delay of $100\mu s$, the encryption phase begins, showing periodic power oscillations (183.8mW average) in time with the elliptic curve point-addition operations before returning to the baseline. Another $100\mu s$ delay leads to the key reconstruction procedure, a subset of the decryption, exhibiting lesser power consumption ($\leq$ 180mW) as computational effort is less than that of key generation. Now follows the decryption phase, characterized by asymmetric power spikes during modular inverse calculations. With the known engineering of the delays to produce valleys between operational peaks, phases could be isolated by timing to discover that power signatures leak algorithmic workflows entirely-a noteworthy side-channel threat where repeated spike patterning (5ms intervals during encryption) could be utilized for key extraction operations.
This pair of graphs depicts the influence exerted by the message size and CPU clock speed on encryption and decryption speeds. The degree of correlation between encryption and decryption times was strongly linear at both 160 MHz and 240 MHz of clock speed. This is expected, as in both instances, the actual choice is XOR operations between the message and key; however, in most cases, encryption will take longer because of the extra overhead of key generation.The decryption is faster since it requires just the reconstruction of the key and not the generation of a completely new one.

Comparing encryption at two clock speeds revealed that the speed performance of cryptographic operations considerably improved when the speed changed from 160 MHz to 240 MHz: for example, when encrypting a 2000 bytes long message, it took 580 μs at 160 MHz but only 380 μs at 240 MHz. The percentage improvement can be calculated as

$$\frac{\text{encryption time (160 MHz)} - \text{encryption time (240 MHz)}}{\text{encryption time (160 MHz)}} \times 100$$

This percentage improvement of 34.45% shows how the higher clock speed favors the performance of cryptographic operations.
In a four-board cryptographic chain, such performance data exhibits minimum overheads in passing encrypted data through various devices. The performance of boards 2 and 3 was almost equal, which contrasts with that of boards 1 and 4. This divergence is, in all probability, the effect of differing power sources: boards 2 and 3 were powered by a well-regulated power supply, whereas power to boards 1 and 4 was provided through USB. These two boards would appear to have been carrying out more complex processes, which certainly would have increased latency, such as transmitting serial data and maybe some WiFi operations. These differences in performance should hardly be considered as overheads from the encryption/decryption chain. Power consumption measurements during cryptographic processes further strengthen the link between the computational complexity and energy usage. The key generation phase occurs as a short-lived, sharp power peak between two 100 μs idle delays. The power increase is small but fully discernible from the baseline level. Encryption Trace: Encryption generates a longer and more complicated trace compared to keygen. The duration and shape of the power pattern during encryption depend on the message size, which means that power consumption increases with the data under treatment. Decryption shows traces of a shorter duration than encryption, while tracking a broadly similar power profile. The statement alleges accepting the testimony of the former due to the existence of a charge, hence the former is indeed less computationally intensive than the latter but with non-trivial processing involved. Conversely, the composite power trace of the full cryptographic routine shows clear differentiation between the key generation, encryption stage, and decryption stage. With each step clearly visible at varying power levels, it asserts that the computational complexity is directly proportional to the instantaneous power consumption.

The practical verification found that the time complexity of cryptographic operations remains highly linear regarding message size (16–2000 bytes) for both single-board and multi-chain configurations; increasing clock speeds from 160 MHz to 240 MHz results in significantly better performance (encryption gets 34.45% faster), whereas sublinear scaling points to other bottlenecks besides the CPU. The power analysis distinguishes them clearly with their operational fingerprints (key generation peaks at 192.8 mW and encryption oscillates at 183.8 mW), therefore corroborating the winning computational intensities alongside

the theoretical ones, yet this also reveals exploitable side-channel patterns. However, multi-board settings show that power stability (externally powered nodes show $\sigma = 5\mu s$ versus USB-powered $\sigma = 18\mu s$ latency variance) and I/O contention dominate real-world performance way more than algorithm overhead. These results collectively affirm the post-quantum viability of isogenies for edge devices, while also pointing to critical engineering trade-offs for clock speed versus power efficiency. The scheme presented in this paper is cryptographically established through the arithmetic geometry and p-adic analysis interface and post-quantum cryptography, using the Engel expansions of Laurent series over $\mathbb{Q}_p$-the field of p-adic numbers. The principle behind the encryption system lies in the representation of Laurent series, formally power series that allow finitely many non-positive exponents, as sequences whose coefficients are then decomposed via the Engel expansion. The Engel expansion succinctly describes a rational number q in $\mathbb{Q}_p$ as a sum of reciprocals of shorter strictly increasing integers, uniquely encoding the structure of q as a sparse additive representation. The decomposition, by far, is best suited to encode highly non-Archimedean algebraic data within the topology of $\mathbb{Q}_p$, providing a representation that is both compact and numerically stable for constrained hardware computation.

Real workings of encryption are bound intimately to supersingular elliptic curves, which are curves defined over finite fields $\mathbb{F}_{p^2}$ having their endomorphism rings isomorphic to maximal orders in quaternion algebras. The cryptographic protocol depends on the computational intractability of finding isogenies between such curves, a problem that holds its ground even on the advent of a quantum adversary. Concretely, in practice, the isogenies are constructed by Vélus formulas, which give explicit isogeny maps if the kernel subgroup is known. The Engel-expanded Laurent coefficients are then used to encode torsion points that will serve as kernels for these isogenies and, thereafter, the encryption keys are put into an algebraic sequence. The lifting of Laurent series into $\mathbb{Q}_p$ via Hensel's lemma serves to carry out stable local expansions around singularities; treating the series as analytic objects with convergence-related properties over the p-adic topological structure. p-adic representations, formed via a field element break into base-p type expansions, are not only a numerical computational device but also a base for encoding isogeny paths that do not factor easy-to-reverse operations. From this point of view arise torsion-subgroup structures on elliptic curves over $\mathbb{F}_{p^2}$ and choices of corresponding 2-torsion points with a non-trivial kernel as the security ingredients of the protocol: the underlying isogeny graph is exponentially large, and extracting the correct path given the public curve is known to be isogeny-path-finding hard, a problem that is assumed to be quantum-resistant.

By incorporating Engel expansions, which inherently are one-way due to their greedy algorithmic construction, this protocol avoids discrete logarithm structures or integer factoring problems known to be solved in polynomial time under Shor's algorithm. Instead, the hardness of inverting the Engel expansion of a Laurent series over $\mathbb{Q}_p$ corresponds to the hardness of reconstructing the isogeny path or the secret torsion point used in the kernel. Hence, the cryptosystem gains semantic security under assumptions that are otherwise quite distinct, and

presently not broken, even by quantum computing models. Implementationally, the system was placed on ESP32 microcontrollers and witnessed an efficient encryption and decryption in constrained environs. p-adic arithmetics here are demonstrated to be computationally feasible and structurally advantageous in real life, whereas usually considered an abstract number-theoretic domain. The scheme is modular in nature and can be varied in terms of p, either the torsion subgroup or the degree of the Engel expansion, to balance between stronger cryptographic manipulations and heavier computation. The entropy in the p-adic representation and the Engel coefficients provides confusion and resistance to side-channel attacks, as hinted in the power consumption traces.

5 Conclusion

This paper bridges cryptography theory and embedded systems to implement a quantum-resistant isogeny-based encryption built from Engel expansions of Laurent series in a p-adic environment to resist Shor's implementation on ESP32 microcontrollers constrained by resources. In this manner, we observe how this line of work may open up both mathematical research and applied research directions, one generalizing the Engel expansion to multivariate Laurent series, thus allowing the development of multivariate isogeny-based cryptosystems with greater key diversity; the other includes analyzing the p-adic dynamical systems induced by iterative isogenies and studying their ergodic properties for considerations of cryptographic randomness and pseudorandom generation. Further, the hardness assumptions behind Engel-inverted Laurent series in $\mathbb{Q}_p$ should perhaps be formalized within the quantum random oracle model, potentially delivering a new domain of cryptographic problems set in local analytic fields. Finally, the arithmetic and geometric methods developed could find further applications beyond cryptography, including coding theory over p-adic fields, p-adic Hodge theory or symbolic computation of rational functions in modular forms. This confluence of number theory and practical embedded security illustrates the changing landscape of post-quantum cryptography, implying that what may seem a distant mathematical isogeny construction is relevantly at the very center of the quantum-resistant technology.

References

[1] De Feo, L., Jao, D. (2011). Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology.

[2] National Institute of Standards and Technology (NIST). (2022). Post-Quantum Cryptography Standardization.

[3] Costello, C., et al. (2016). Efficient algorithms for supersingular isogeny Diffie-Hellman. CRYPTO.

[4] Koziel, B., et al. (2017). SIKE on ARM: Embedded post-quantum cryptography. CHES.

[5] Schweiger, F. (1995). Multidimensional continued fractions and Engel expansions. Bulletin of the London Mathematical Society.

[6] Koblitz, N. (2012). p-adic numbers, p-adic analysis, and zeta-functions. Springer.

[7] Kedlaya, K. (2006). p-adic differential equations. Cambridge University Press.

[8] Liu, A., Ning, P. (2008). TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. IPSN.

[9] Oder, T., et al. (2018). Practical CCA2-secure post-quantum encryption on embedded devices. CHES.