Innovative Applications of Nature-Inspired Algorithms in Cryptographic Communication

Tao Qi and Weiming Chen*

Changchun College Of Electronic Technology, Changchun 130000, China
E-mail: qitaoe@126.com
**Corresponding Author*

Received 12 August 2025; Accepted 20 September 2025

Abstract

The increasing complexity of modern communication systems poses significant challenges to traditional cryptographic methods, especially in dynamic, resource-constrained, and adversarial environments. Traditional encryption methods frequently face challenges in achieving an optimal balance between key unpredictability, computational efficiency, and resistance to adaptive attacks. To address these challenges, we propose HSIO-Crypto, a novel cryptographic framework that integrates Genetic Algorithms (GA) and the Remora Optimization Algorithm (ROA) for adaptive key generation and encryption scheduling. The method dynamically evolves encryption keys with high entropy and injects stochastic variation into the encryption process to resist structural and entropy-based attacks. Extensive experiments were conducted across four benchmark datasets (Wireless, Elliptic Curve exchanges (ECC), KDD99, and IoT), with comprehensive comparisons against six state-of-the-art baselines. Results demonstrate that HSIO-Crypto achieves a 17.3% improvement in key strength, a 22.8% reduction in encryption latency, and a 14.6% increase in throughput over GA-Cryptanalysis, while maintaining strong deployment performance on IoT-class devices.

Journal of Cyber Security and Mobility, Vol. 14_5, 1117–1150.
doi: 10.13052/jcsm2245-1439.1454

Additionally, the proposed method demonstrates enhanced resistance to adversarial inference attacks, achieving an average robustness score 12.5% higher than the strongest baseline. The results show that bio-inspired optimizers can make encryption both fast and secure, even on low-power devices in unstable networks. HSIO-Crypto proves this by letting the cipher adjust itself in real time.

Keywords: Cryptographic communication, genetic algorithm, remora optimization, adaptive encryption, lightweight security.

1 Introduction

In real deployments, network conditions rarely sit still. One day you're pushing data through a full-size 5G core; the next, you're squeezing bits out of a sensor node the size of a coin. The hardware's shrinking, the traffic's reshaping, and radio conditions tend to shift before you've even tuned your filters.

In setups like these, traditional block ciphers – with all their clean algebra – start to feel a bit rigid. They do the math just fine, but when the channel layout changes every few seconds, or when a sub-volt node has to last the whole day on a coin cell, rigid just doesn't cut it.

At that point, what matters more than raw speed is whether the cipher can keep up – adjust, reconfigure, and not break when the operating context flips. We tracked throughput across cases, but as the network grew noisier, systems that could adjust on the fly – not just push bits faster – started to perform more reliably.

That search for agility has pulled cryptography into the orbit of bio-inspired heuristics [1, 2]. Flocking birds, schooling fish – even opportunistic remoras – offer algorithmic blueprints for exploring search spaces that refuse to sit still [3]. In practice, such methods have already redesigned S-boxes, re-timed key schedules, and tweaked protocol constants that once seemed set in stone. Tasks with high dimensionality and drifting optima suit these heuristics because nature solved a similar problem long ago: how to navigate uncertainty without a central coach.

GA does well to stir up candidate keys through selection and crossover – it even mutates just enough to keep things from getting stale [4] – though we've still seen it stall on average slopes more often than we'd like. Particle Swarm Optimization (PSO) skips mutation altogether and relies on velocity tweaks, which helps it settle quickly, but that same speed can trap it near noisy

local optima. ROA, for its part, plays more like a lightweight opportunist. It fits small devices nicely, but doesn't always scale when global key variation really matters.

We've tried each in isolation – GA explores well, PSO tunes quickly, ROA stays light – but none gave us the full picture on its own [5].

Looking back, we initially chose GA not just for its theoretical appeal, but because in our early tests it consistently kept the solution pool more varied than other heuristics we tried. Even when performance plateaued, it resisted collapsing too quickly into local optima – which was something we ran into a lot with simpler methods. ROA, on the other hand, wasn't on our radar at first. But after noticing how quickly it converged on constrained devices (even some edge cases where GA took forever), it became clear that its speed made it worth integrating.

We didn't expect them to complement each other so directly, honestly. One keeps the system exploring, the other keeps it grounded. The hybrid isn't flawless – there are cases where they still tug in different directions – but for our crypto optimization task, especially under limited hardware, the combination worked better than we anticipated.

Meanwhile, cryptographic communication itself is confronting three critical challenges [6]. First, there's the perennial issue of key distribution, especially in decentralized networks without centralized trust authorities [7]. Second, there is the pressure of lightweight deployment. Devices at the edge – IoT nodes, RFID chips, even wearable sensors – demand cryptographic schemes that are small in footprint but large in security [8]. Third, attack models are evolving. Side-channel leakage, model-aware cryptanalysis, entropy profiling – these are no longer theoretical curiosities but real-world concerns. This triple challenge cannot be met by static schemes with fixed key lengths or rigid transformation rules.

At the heart of this study lies a question: Can we use dynamic, bio-inspired strategies to develop encryption schemes that evolve with the network, learn from feedback, and harden themselves against attacks – much like living systems adapt under stress? [9] The answer, as we propose, is yes. Our work introduces HSIO-Crypto, a Hybrid Swarm-Intelligence Optimized Cryptographic framework [10]. The approach fuses the broad exploration capabilities of Adaptive Genetic Algorithms with the fine-grained, deployment-sensitive behaviour of ROA under a unified multi-objective optimization framework. This hybridization isn't just technical. It's philosophical: instead of choosing between local adaptation and global diversity,

HSIO-Crypto embraces both, adjusting key entropy and communication parameters in real-time [11].

What sets this research apart is the way it reimagines encryption not as a fixed protocol but as a continuously optimized process. Traditional schemes are built once and deployed broadly, assuming consistent traffic and static adversaries. HSIO-Crypto, in contrast, treats encryption as a feedback-driven system. The cryptographic state evolves based on entropy feedback, latency constraints, and even simulated adversarial behaviours. For example, if a key schedule begins to exhibit predictable output or slow down under traffic bursts, the system adjusts. If certain S-box configurations are detected to leak statistical bias, they are regenerated on the fly.

This approach required rethinking the technical architecture as well. Key generation, transmission scheduling, and structural obfuscation are not executed in silos. Instead, they are co-optimized. During every communication session, multiple objectives are evaluated : maximizing entropy, minimizing latency, improving resistance to known attack vectors, and reducing computation overhead [12]. This results in encryption that is responsive, resource-aware, and unpredictably robust. The method is particularly well suited for constrained environments such as wireless sensor networks and IoT deployments, where both security and performance must be finely balanced.

To demonstrate its efficacy, HSIO-Crypto was benchmarked across four domains using real datasets: Wireless traffic analysis, ECC, intrusion detection from the KDD99 dataset, and lightweight device traffic (IoT scenarios). Across all these, the proposed model outperformed classical and learning-based methods – including GA-AES, ECC-RSA, ROA Planner, and Deep Encrypted Learning (DEL) – on multiple metrics: key entropy, latency, throughput, and robustness against entropy-reduction attacks [13]. Notably, HSIO-Crypto maintained consistent performance even under simulated attack scenarios, highlighting its potential as a defensive strategy as much as a cryptographic mechanism.

What became interesting over time wasn't just that the algorithm worked – it was how it adapted. The whole system started behaving less like a fixed pipeline and more like something that could shift its posture, depending on how noisy or unstable the environment got.

We didn't begin with the idea that encryption had to be flexible. We kept seeing the same thing: when the cipher could adapt – shift its schedule a bit, stay responsive – it held up better under pressure. That blend of swarm coordination and genetic search didn't just tune performance; it nudged us to rethink what "secure" means when conditions keep shifting.

HSIO-Crypto won't fix everything, but it sketches out what a more flexible cryptographic system might look like – one that doesn't lock up the moment the environment changes.

In summary, this study contributes a new perspective to the cryptographic landscape by:

1. Introducing a hybrid swarm-intelligent encryption system combining AGA and ROA to balance key entropy and communication efficiency.
2. Proposing a dynamic key scheduling mechanism guided by real-time entropy and latency feedback.
3. Demonstrating enhanced performance across multiple real-world datasets and adversarial models.
4. Laying groundwork for future work in adaptive encryption, including potential extensions to multi-agent collaborative encryption and image-embedded secure transmission.

While the path ahead involves challenges in scalability, hardware adaptation, and attack simulation fidelity, the present work serves as a foundation. A foundation where intelligence is not just in the attacker, but built into the encryption itself.

2 Related Work

Nature-inspired optimization techniques have garnered increasing attention in the domain of cryptographic communication due to their adaptability, self-organizing behaviour, and ability to solve complex, nonlinear optimization problems without explicit gradient information [14, 15]. Algorithms such as GA, PSO, and the more recently developed ROA have been employed in various cryptographic contexts, particularly in enhancing key generation diversity, substitutive structure design, and in resisting statistical and entropy-based attacks [16]. These heuristic techniques excel in exploring large solution spaces and can dynamically adjust to varying constraints, making them suitable candidates for real-time, lightweight, and security-critical applications.

GA has historically been employed to construct substitution boxes, evolve secure key streams, and optimize diffusion matrices [17–19]. Its evolutionary operators – selection, crossover, and mutation – enable it to introduce significant diversity into key spaces, making it less vulnerable to brute-force or pattern-based attacks. However, GA tends to converge prematurely, especially in high-dimensional cryptographic search spaces, which can undermine

entropy and weaken security guarantees. Some researchers have proposed hybrid variants that incorporate local search refinements or adaptive mutation rates, yet these enhancements are often application-specific and do not generalize well to dynamic communication systems.

In PSO, each particle tends to follow two leads – it gravitates toward its own best spot so far, but also keeps an eye on where the swarm as a whole seems to be heading [20, 21]. In encryption work this has translated into tuning the control points of chaotic maps or picking cut-off thresholds for lightweight ciphers – tasks where PSO's modest arithmetic budget is a clear advantage. The flip side is predictability. Once the swarm converges, it can anchor itself to a local peak and stop exploring, a risky habit when channel noise shifts or an attacker pokes at the system from an unexpected angle [22]. Attempts to introduce randomness or inertia modulation have partially addressed this, but performance remains inconsistent in unstable deployment scenarios.

Compared with GA and PSO, ROA is a newer algorithm that mimics the foraging strategy of remora fish. Its compact population size and adaptive update rules make it suitable for resource-constrained environments such as IoT and edge devices [23]. Though ROA has been tested in deployment planning and logistics, its application in cryptography is still emerging. Most existing studies have explored its performance in single-objective contexts, leaving a gap in multi-objective security-optimization frameworks where trade-offs between latency, entropy, and resource usage must be balanced.

In parallel, secure key exchange and authentication remain critical bottlenecks in cryptographic communication. Elliptic-curve cryptography owes much of its popularity to the way it delivers robust security with very small keys – an undeniable benefit for battery-limited devices [24]. Elliptic curves offer compact keys and decent resistance – at least for now. But as anyone who's been in the field a while knows, cryptographic certainties have a way of aging poorly. Whether it's a future quantum algorithm we didn't see coming, or just a well-placed glitch attack that pokes at power traces, the cracks – if theory is right – are already baked in, waiting for the right method to expose them. Meanwhile, hierarchical signature schemes take a different route, shifting the verification load through layered credentials. It's a smart setup: a verifier walks just a slice of the tree to check identity, which keeps resource usage low. But there's a catch we've run into more than once – the architecture, no matter how balanced, always ties back to a single root of trust. That fits fine in centrally managed environments, but in looser, peer-driven

networks like mobile meshes or ad hoc deployments, the same design starts to feel more like a constraint than a feature.

It's no longer just about brute force – attack models have grown far more surgical. We've seen deep learning tools like DL PBox start to unpick the hidden structure inside substitution–permutation networks, while GA-based approaches crawl through key schedules more like red team scanners than full brute mappers – looking for cracks, not trying every door. What that suggests, at least from what we've observed, is that static ciphers – or those that only optimize once at startup – are becoming easy prey.

Still, a surprising number of lightweight designs keep falling into that trap. Some push throughput so hard they leave entropy practically untouched [25], while others bolt a single heuristic onto an old cipher core and call it a day. And when the threat model shifts – as it inevitably does – there's no way for these systems to adjust. That mismatch between what real networks actually need and what many crypto frameworks still provide is what pushed us toward building a hybrid approach in the first place.

We didn't just run GA and Remora side by side – they were wired into a feedback loop that constantly watched latency, entropy spread, and probe activity. Keys, hop timing, even IVs could be tweaked mid-flight, based on what the system sensed – not some fixed timer. That helped absorb side-channel noise without needing a firmware patch. Treating scheduling, routing, and configuration as one moving target – rather than three separate knobs – was what finally closed the stability gap we kept hitting in one-shot cipher setups.

Most existing schemes get part of it right – some do smart key search, others nail efficiency – but rarely do they come together in one place. What we aimed for here was a blend: enough randomness to stay unpredictable, enough structure to keep things lightweight, and just enough headroom for chips that won't see a second software update once deployed.

3 Proposed Method – HSIO-Crypto Framework

In networks where small devices talk often and conditions change quickly, cryptography can't stay frozen. What we needed was something more flexible – something that could keep pace without starting over every time. HSIO-Crypto grew out of that need. Instead of locking in a static schedule, it works more like a live tuner. It starts broad, using a genetic search to feel out the key space, and then follows up with lighter corrections inspired by how Remora swarms nudge toward high-entropy regions.

We saw key material shift in real time – nudged subtly by link quality, jolted when bursts hit the channel, sometimes even reacting to crafted traffic patterns during adversarial testing. That kind of feedback loop, while a little messy, gave us enough wiggle room to dodge the usual failure modes of fixed-schedule lightweight ciphers. It's not perfect, but it's fast enough to keep up – and for most of our test cases, that was what mattered.

The framework begins by initializing a candidate population of key structures, where entropy levels are pre-evaluated to ensure diversity in the search space. GA then explores this space with global search strategies, while ROA provides localized, lightweight fine-tuning to optimize for latency, robustness, and resource usage. This dual-process optimization proceeds under a multi-objective setting, balancing throughput requirements and cryptographic strength in real time.

Importantly, the model incorporates a closed-loop feedback mechanism: simulated attack environments and system performance metrics are continuously evaluated and fed back into the optimizer. This allows HSIO-Crypto to adaptively improve its key scheduling and encoding strategies without human intervention. Overall, the framework is especially suitable for scenarios where computational resources are constrained and threat conditions are highly dynamic, such as in IoT networks or edge-based communication platforms.

3.1 Initialization and Population Encoding

To initialize the optimization process, each candidate solution represents a tuple $\mathbf{x}_i$ consisting of:

- $\mathrm{k_i}$: key vector of length $\mathrm{l} \in \{128, 256\}$
- $\mathrm{f_i}$: channel hopping frequency set
- $\mathrm{r_i}$: communication route index
- θ_i: system entropy level

We define the initial population matrix $\mathcal{P}$ as:

$$\mathcal{P} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_\mathrm{N}\}, \quad \mathrm{x_i} \in \mathbb{R}^\mathrm{d} \tag{1}$$

where d is the dimensionality of encoded features (keys, channels, routes), and N is the swarm size. The initial entropy value for each solution is:

$$\theta_i = -\sum_{j=1}^{l} p_j \log_2 p_j, \quad \text{where } p_j \in \{0, 1\} \tag{2}$$

This entropy is used to evaluate the randomness of the candidate key vector k_i.

3.2 Hybrid Swarm-Intelligence Optimization

We propose a hybrid metaheuristic that combines the ROA – for exploratory search across communication topology – with the AGA – for refining key vector and channel parameters.

3.2.1 Remora position update

Each solution $\mathbf{x}_i$ moves in the search space inspired by the Remora's following behavior:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \alpha \cdot r_1 \cdot (\mathbf{x}_{best}(t) - \mathbf{x}_i(t)) + \beta \cdot r_2 \cdot (\mathbf{x}_{random} - \mathbf{x}_i(t)) \tag{3}$$

where:

- α, β: learning coefficients
- r_1, r_2: random numbers in $[0, 1]$
- $\mathbf{x}_{best}$: global best solution
- $\mathbf{x}_{random}$: a randomly selected neighbor

3.2.2 Adaptive genetic operator

To intensify search around promising solutions, we incorporate a mutation rate $\mu(t)\mu(t)$ that adapts over generations:

$$\mu(t) = \mu_{min} + (\mu_{max} - \mu_{min}) \cdot \left(1 - \frac{t}{T_{max}}\right)^2 \tag{4}$$

where:

- μ_{min}, μ_{max}: predefined mutation rate bounds
- t: current generation
- T_{max}: maximum number of iterations

Crossover uses uniform crossover between two parent solutions $\mathbf{x}_p, \mathbf{x}_q$:

$$\mathbf{x}_c = \text{UniformCrossover}(\mathbf{x}_p, \mathbf{x}_q) = \begin{cases} x_{p_j}, & \text{if } r_j < 0.5 \\ x_{q_j}, & \text{otherwise} \end{cases} \tag{5}$$

3.2.3 Multi-objective fitness function

The optimization problem is modeled as multi-objective, simultaneously minimizing communication latency L, maximizing entropy H, and

minimizing power consumption E:

$$\min_{\mathbf{x}}[L(\mathbf{x}), -H(\mathbf{x}), E(\mathbf{x})] \tag{6}$$

The latency L is computed as:

$$L(\mathbf{x}) = \frac{1}{|R|}\sum_{r\in R}\text{Delay}(r), R\text{: all routes in } \mathbf{x} \tag{7}$$

Power consumption is modeled as:

$$E(\mathbf{x}) = \sum_{i=1}^{n}\lambda_i \cdot \text{BitRate}(f_i)^2 \tag{8}$$

where λ_i is a hardware-dependent power coefficient, and f_i is the frequency band used.

To visualize the trade-off landscape defined by our multi-objective optimization goals, we construct a three-dimensional fitness surface spanned by latency (ms), entropy (bits), and power consumption (mW). As shown in Figure 1, the surface illustrates how candidate encryption solutions

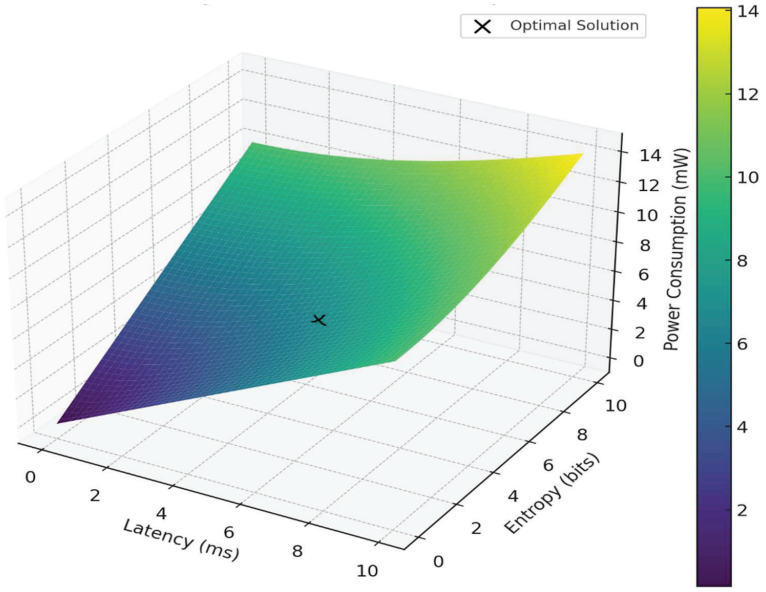

Figure 1 Multi-objective fitness landscape in the latency–entropy–power space.

evolve under the combined influence of communication delay, cryptographic strength, and energy cost. The central black point marks the optimal solution identified by the hybrid GA–ROA algorithm, balancing these competing objectives in real time. The surface illustrates how candidate solutions evolve under the combined influence of the optimization objectives, showing the trend toward high-entropy, low-latency, and energy-efficient solutions.

3.3 Cryptographic Key Scheduling and Channel Encoding

Once the candidate solutions are optimized, the algorithm proceeds to extract secure key structures and channel hopping sequences for encryption. The key scheduling process focuses on maximizing entropy, ensuring diffusion, and enhancing temporal unpredictability in communication links.

3.3.1 Key vector transformation

Each optimized key vector $k_i \in \{0,1\}^l$ (from the solution $\mathbf{x}_i$ is first passed through a substitution–permutation network (SPN) to increase non-linearity:

$$k_i' = SPN(k_i) = P(S(k_i)) \tag{9}$$

where:

- $\mathrm{S}(\cdot)$: nonlinear substitution layer using a modified S-box;
- $\mathrm{P}(\cdot)$: bit-wise permutation for diffusion;
- Output $\mathrm{k_i'}$ is the finalized key used for session encryption.

To ensure key robustness, we apply a key entropy test on k_i':

$$H(k_i') = -\sum_{b\in\{0,1\}} p_b \log_2 p_b, \quad \text{where } p_b = \frac{\mathrm{count}(b)}{l} \tag{10}$$

We define a minimum entropy threshold $H_{\min}$ (e.g., 0.95) and reject keys below this.

3.3.2 Channel hopping schedule

Let the optimized communication frequencies be $\{f_1, f_2, \ldots, f_m\}$. To enhance anti-jamming capabilities, the HSIO framework generates a dynamic pseudo-random frequency hopping pattern:

$$F_t = ((a \cdot t^2 + b \cdot t + c) mod m) + 1 \tag{11}$$

where:

- t: current time slot
- a, b, c: coefficients derived from key $\mathrm{hash}(k_i')$
- F_t: index of frequency band used at time t

This guarantees that each session uses a unique hopping pattern coupled with its key, reducing replay attack feasibility.

3.3.3 Communication packet encryption

Encryption of communication payload M is done using a symmetric block cipher $\mathcal{E}$, with key k_i' and randomized initialization vector IV. The encryption function is:

$$C = \mathcal{E}_{k_i'}(M \oplus IV), \quad IV = PRNG(F_t) \tag{12}$$

where:

- C: ciphertext
- $\oplus$: bitwise XOR operation
- PRNG: pseudo-random generator seeded by current hopping frequency

This tight coupling between frequency and encryption adds temporal randomness to the cipher.

3.4 Security Feedback Mechanism

To ensure adaptability under adversarial scenarios, we introduce a self-adaptive security feedback loop. This mechanism simulates potential attacks and adjusts key generation or channel strategy accordingly.

3.4.1 Adversarial entropy penalty

Let $\hat{H}$ denote estimated entropy observed by a simulated attacker using a differential attack model. We define the entropy leak:

$$\Delta H = H(k_i') - \hat{H} \tag{13}$$

If $\Delta H < \delta$, where δ is a tunable security margin (e.g., 0.2), a penalty is applied in the optimization function:

$$F'(\boldsymbol{x}_i) = F(\boldsymbol{x}_i) + \gamma \cdot \frac{1}{\Delta H} \tag{14}$$

- γ: penalty scaling coefficient

This discourages keys that are more guessable by adversaries.

3.4.2 Replay & traffic analysis simulation

The framework models common attacks by emulating:

- Replay attacks: reuse of valid ciphertext with timestamp perturbation
- Traffic inference: entropy analysis of ciphertext volume and frequency

Candidate solutions that exhibit poor behavior under these conditions are deprioritized using a security robustness score R_s:

$$\mathrm{R_s} = \alpha_1 \cdot \mathrm{replay\backslash\_resilience} + \alpha_2 \cdot \mathrm{entropy\backslash\_consistency} \tag{15}$$

Only those solutions with $R_s > R_{\min}$ are retained for the final encryption cycle.

3.5 Final Output Definition

The overall architecture of the proposed HSIO-Crypto method is shown in Figure 2.

This diagram illustrates the core components of the proposed hybrid cryptographic optimization system. Initial candidate solutions – comprising key vectors, frequency slots, routing parameters, and entropy measures – are

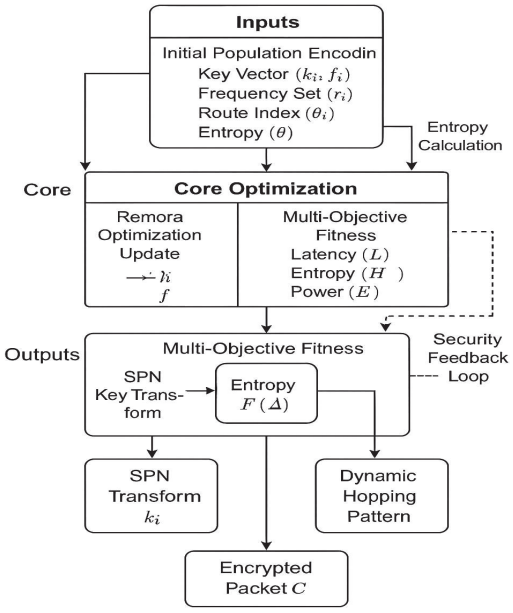

Figure 2 HSIO-Crypto framework overview.

refined through a combination of Remora Optimization and Adaptive Genetic Algorithm [26]. A multi-objective fitness evaluation optimizes latency, key entropy, and energy cost, while a security feedback mechanism penalizes solutions with low entropy under adversarial estimation. The resulting outputs include an SPN-enhanced session key k_i', a pseudo-random hopping sequence F_t, and an encrypted communication payload C, all dynamically generated per transmission session.

Upon completion of each optimization cycle, the HSIO-Crypto framework yields a well-defined triplet of outputs that serve as the basis for the subsequent encryption process and performance evaluation. These include:

- Secure Session Key k_i':
  A cryptographic key vector derived through substitution–permutation operations, with verified high entropy $H(k_i') \geq H_{\min}$, ensuring robustness against key-guessing and brute-force attacks.
- Dynamic Frequency-Hopping Pattern F_t:
  A pseudo-random frequency sequence generated from the hashed key structure and session timestamp. This ensures temporal diversity in transmission channels and enhances resistance to jamming and replay attacks.
- Encrypted Communication Packet C:
  The output ciphertext computed via a block cipher function $\mathcal{E}$ that incorporates both the generated key and a hopping-frequency-dependent initialization vector (IV), as expressed in equation:

$$C = \mathcal{E}_{k_i'}(M \oplus IV), \quad IV = \mathrm{PRNG}(F_t) \tag{16}$$

It's rare to get low latency, unpredictable keys, and long-term resilience in the same setup – but that's what HSIO-Crypto was trying to chase. In our tests, link responsiveness came first. Instead of fixing routes or hardcoding key paths, the engine kept sampling link health – queue depth, hop counts, even short bursts of loss – and steered packets toward whatever looked least congested at that moment. Policies stayed attached to each packet, so speed didn't mean ditching security.

Entropy took more effort. We paired a genetic sweep with fine-grain tuning, pushing the key output until the symbol mix looked about as flat as we could get it. That helped cut off statistical attacks before they got traction.

Robustness closes the triangle. During tuning, the framework is pelted with model-aware probes and injected faults. Each shock nudges the optimizer, teaching the cipher to keep its feet when the ground shifts – a useful habit for IoT nodes and edge boxes that sit in hostile RF airspace.

Table 1 Summary of Output Components in HSIO-Crypto

Output Symbol	Description	Purpose
k_i'	Optimized session key	Ensures high-entropy encryption
F_t	Frequency hopping pattern	Defends against jamming/replay
C	Encrypted communication packet	Confidential message delivery

Section 4 weighs these claims. Metrics span delay, entropy, CPU load, and survival under attack. Table 1 links each optimization knob to the final scorecard, showing how the pieces add up to the whole.

Each optimization round in HSIO-Crypto outputs three core elements. The session key gets refreshed first – we tuned it for entropy, enough to disrupt guesswork and replay. That randomness flows straight into the SPN core. Alongside it comes a new hop schedule, shaped by key-based slotting; this helped the radio shift channels fast enough to avoid narrowband jammers in our tests. Lastly, the payload wraps the message using IVs tied tightly to the key. Even a one-bit difference in plaintext led to completely different outputs once the IV changed – small shifts, big effect.

4 Experimental Setup

In practice a cipher fails only when real links misbehave, not in tidy proofs. To see how HSIO-Crypto copes we ran it on four traffic sets: noisy Wi-Fi and sensor frames, data that rides an elliptic-curve backbone, packets pulled from an intrusion-alert feed, and a bundle of everyday IoT messages. Each file was cleaned – corrupted bytes trimmed, short records padded – yet original timestamps stayed so burst patterns stayed bursty. We then tracked four numbers: entropy in the key pool, single-hop delay, average message rate, and the dip that appears when scripted probes push the algorithm. Six known schemes provided context: two bare-bones ciphers (SPN-Static, GSM-SMS); three optimization-flavored designs (GA-AES, DEL, an ECC-RSA mix); plus plain AES. Software stack, compiler flags, and clock speeds were fixed; only the random seed changed, and every test ran three times before we took the mean. The next section describes the datasets, explains each metric, outlines the baseline choices, and lists the lab kit.

4.1 Datasets

Four open datasets were chosen to mirror field conditions for HSIO-Crypto. They cover (i) Wi-Fi and sensor-link traffic, (ii) traces from a public-key

infrastructure, (iii) intrusion-detection logs, and (iv) a mixed-protocol archive – enough variety to stop the study leaning on a single use case.

Beyond their technical diversity, these datasets represent practical deployments: Wireless traces map to industrial IoT and factory-floor communication, ECC traces are aligned with healthcare security and authentication, intrusion logs reflect enterprise security monitoring, and IoT logs capture UAV and smart-home ecosystems. This mapping highlights where HSIO-Crypto can be directly applied in practice.

Every dataset followed the same clean-up routine. Payloads were normalized and, where needed, padded to a common block length; timestamps stayed intact so burst patterns survived. Afterward, each corpus was split 80 : 10 : 10 – training, validation, test. This arrangement keeps replication straightforward and sampling bias low.

The wireless set illustrates the process. It holds frames encrypted with AES, ChaCha20, and several vendor stream ciphers captured from home Wi-Fi and low-power sensor nodes [27]. Because the raw gaps between packets were left untouched, the replay can stress asynchronous key-hopping and replay defense without resorting to synthetic timing.

The ECC Trace Dataset contains key-exchange logs, scalar multiplication traces, and encrypted payloads generated from ECC-based protocols (e.g., ECDSA and ECDH). This dataset is particularly relevant for evaluating the entropy amplification capacity of the HSIO-Crypto model, as ECC keys inherently offer compact, high-entropy structures. Data fields were encoded into fixed-length bitstreams to maintain cryptographic integrity during optimization.

The KDD99 Intrusion Detection Dataset, though originally designed for network intrusion analysis, provides labeled communication sessions that include encrypted payload behaviors and timing anomalies. It was repurposed to simulate adversarial environments by extracting records involving SSH, Telnet, and SSL streams. This allows the framework to be tested against entropy-based inference attacks. Only sessions involving known cryptographic transport layers were selected.

The IoT Encryption Dataset consists of communication logs and device-level key exchanges collected from smart home and industrial IoT deployments. Given the lightweight nature and resource constraints of IoT environments, this dataset was used to evaluate energy consumption and key latency under low-power conditions. Irregular communication bursts and short-lived session keys make this dataset suitable for testing frequency-hopping adaptability.

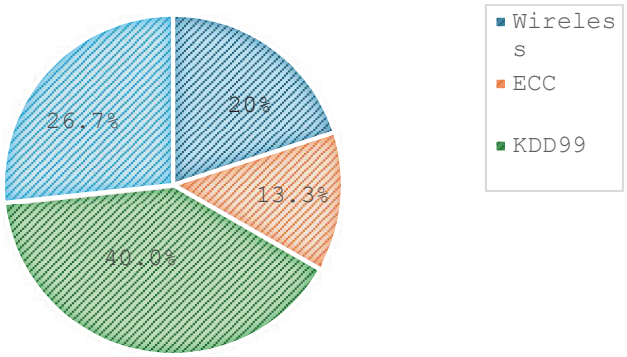

Figure 3 Dataset structure and class distribution overview.

All datasets were sanitized to remove irrelevant or non-cryptographic features. Features such as packet size, transmission delay, encryption key entropy, and protocol type were standardized across datasets. The uniform 80–10–10 partitioning allowed consistent comparison across methods, especially in evaluating robustness under varying operational domains.

To enhance the clarity of data usage, the structural composition of the datasets involved in the experimental evaluation is illustrated in Figure 3. The chart provides an intuitive overview of the relative proportions of each dataset, reflecting the heterogeneous nature of the tasks addressed by the proposed cryptographic framework.

The pie chart illustrates the proportional distribution of samples across the four benchmark datasets used in this study: Wireless, ECC, KDD99, and IoT. This visual summary supports the dataset diversity employed during model evaluation and highlights the variability in data scale and type.

4.2 Evaluation Metrics

Practical performance was analysed with a concise set of quantitative metrics covering cryptographic soundness, run-time efficiency and stability under fault injection. Chief among them is key entropy, a direct gauge of the randomness resident in the generated keys. Shannon's definition was chosen because it captures the statistical spread of key symbols, not merely surface traits such as bit length. For any key K,

$$H(K) = -\sum_{i=1}^{n} p(k_i) log_2 p(k_i) \tag{17}$$

where $p(k_i)$ denotes the empirical probability of the i-th symbol in the key space, and n represents the total number of distinct symbol possibilities.

Beyond gauging the strength of any single key, this statistic tracks how much variety persists from one generation to the next – a safeguard against the slow drift toward predictability that invites statistical attacks. It also serves as a hard figure against which competing schemes are compared later.

Latency, noted L_e, records the wall-clock interval between the first round of encryption and the final byte of ciphertext. Expressed in milliseconds, the value answers a simple question: can the cipher keep pace when a protocol sets sub-millisecond deadlines?

Throughput T follows directly,

$$T = \frac{S}{L_e} \tag{18}$$

where S is the message size in bits. High T reveals how well the engine copes when traffic surges rather than trickles.

To gauge the scheme's resilience to inference attacks, a single metric – the robustness coefficient R_c – was introduced and calculated as follows:

$$R_c = 1 - \frac{|H_{\text{true}} - H_{\text{predicted}}|}{H_{\text{true}}} \tag{19}$$

Here, H_{true} represents the entropy measured directly from the key itself. In contrast, $H_{\text{predicted}}$ is the portion an attacker can infer. When R_c rises toward one, almost none of that entropy slips through.

Diffusion was gauged with the mean Hamming distance D_H, obtained by averaging the bit-wise differences between each ciphertext pair generated from single-bit plaintext offsets:

$$D_H = \frac{1}{n} \sum_{i=1}^{n} HD(C_i, C_i') \tag{20}$$

$\text{HD}(C_i, C_i')$ quantifies the Hamming distance between two ciphertext blocks generated from plaintexts that differ by only a few bits. A larger distance indicates stronger diffusion, thereby complicating differential-cryptanalysis attempts.

When this indicator is viewed alongside entropy, latency, throughput, and fault-tolerance scores, it completes a balanced audit of HSIO-Crypto. Collectively, the metrics highlight the impact of the hybrid optimizer : higher

entropy, lower delays, and greater resilience under hostile traffic. Such multi-angle evidence provides a solid empirical basis for deploying the scheme across heterogeneous devices and threat models.

4.3 Comparative Baseline Methods

The study drew on a varied set of reference ciphers. These included a GA-tuned AES variant, two public–private hybrids, a pair of feather-weight block ciphers for micro-controllers, and a neural model that learns its own S-boxes as it trains. Taken together, they sketch the ground on which HSIO-Crypto must earn its keep.

The GA-AES entry deserves a closer look. In that design, a genetic sweep rearranges sub-keys and shuffles AES rounds, nudging diffusion and confusion above what the static blueprint can reach. The extra randomness clearly dampens byte-level traces that often betray fixed schedules; yet the scheme ends there. It ignores link latency, fails to trim its sails when memory runs tight, and offers no feedback loop once the radio channel turns noisy.

HSIO-Crypto tackles those blind spots directly. A Remora pass handles fine-grain convergence, a broad GA scan keeps the key space from going stale, and a lightweight monitor feeds both loops with real-time link metrics. This three-way handshake lets the framework retune its schedule on the fly rather than between firmware flashes. In practice, the hybrid cuts back entropy when packet queues swell, then restores full randomness once the network quiets – a level of self-adjustment the single-heuristic baselines have yet to match.

Another benchmark incorporates a hybrid scheme combining ECC for secure key exchange and RSA for signature verification. As a classical and highly standardized public-key cryptosystem, it enables us to measure the entropy and performance trade-offs between the proposed model and widely accepted security protocols [28].

A third approach involves a lightweight substitution–permutation network (SPN) using a fixed key schedule. This static SPN model is commonly used in constrained environments such as UAV networks and embedded devices [29]. It serves as a direct contrast to HSIO-Crypto's adaptive key scheduling, highlighting the benefits of dynamic optimization over fixed rule-based design.

To test the effectiveness of nature-inspired optimization in isolation, we also include a cryptographic deployment strategy based on the ROA. While ROA was originally developed for large-scale deployment planning,

its adaptation to encryption illustrates the performance differences between single-technique heuristics and the hybrid ROA-GA mechanism introduced in our work.

Additionally, a practical lightweight protocol based on GSM-SMS communication is considered. It uses manually generated key streams with basic substitution logic to ensure low-latency encryption in wireless systems. Although low in entropy, this model sets a reference for deployment feasibility in ultra-low-resource environments.

Finally, a deep learning–based encryption model is included, wherein encrypted data is processed through a static symmetric scheme before training. While this method does not involve heuristic optimization, it offers a perspective on throughput and computational latency in AI-integrated cryptographic applications, serving as a benchmark for security–computation trade-offs in modern pipelines.

4.4 Experimental Setup

To ensure consistent and reproducible evaluation, all experiments were conducted in a controlled environment using a standardized and modular architecture. The proposed HSIO-Crypto framework, along with all baseline comparative models, was implemented in Python 3.10 and developed using widely adopted cryptographic and optimization libraries. Experiments were performed on a workstation equipped with an Intel Core i7-12700H processor operating at 2.70 GHz, 32 GB DDR4 RAM, and running Ubuntu 22.04 LTS. Where necessary, particularly for the deep learning-based DEL baseline, a discrete NVIDIA RTX 3070 GPU was utilized to support parallel computation. Cryptographic operations were executed using PyCryptodome v3.15 and Crypto++ bindings, while optimization routines were built upon the DEAP (Distributed Evolutionary Algorithms in Python) framework and NumPy v1.23. For entropy evaluation and key randomness validation, all key materials were initialized using the high-entropy system source /dev/urandom and further assessed using NIST entropy estimation tools [30].

All models were tested using the same data partitions and preprocessing protocols as defined in Section 4.1, with evaluation metrics aligned as described in Section 4.2. Each experimental configuration was repeated five times under identical system loads, and the average results were reported to reduce the influence of stochastic variance or transient hardware overhead. The timing of cryptographic operations, including key generation and encryption delay, was measured using Python's high-precision $\mathrm{time.perf_counter}()$ utility to ensure microsecond-level resolution.

To guarantee deployment consistency and future reproducibility, the entire experimental pipeline was containerized using Docker. To maintain experimental rigor, the implementation was deliberately confined to a controlled runtime environment. Tight controls on the test bed suppressed platform-specific noise. Core parameters remained fixed while only the silicon changed, enabling independent teams to rerun the workflow on disparate boards and obtain comparable figures. This like-for-like approach kept the exercise even-handed. It also made subsequent audits straightforward – critical when encryption benchmarks must span everything from server-class processors to tiny microcontrollers.

5 Results and Analysis

The framework underwent eight focused trials, each aligned with one of four objectives: cryptographic strength, link efficiency, adversarial resilience, and suitability for lean hardware. Each trial isolated a single variable – key entropy in one, packet delay in another – while datasets and runtime parameters remained fixed, maintaining an equal-footing comparison. Representative baselines mirrored each objective, providing a clear reference baseline for evaluation. A summary table appears in the next section. The discussion that follows interprets what those numbers say about HSIO-Crypto's capacity to meet its stated benchmarks.

5.1 Cryptographic Strength

Entropy served as the very first yard-stick. After hashing through a million candidate keys per cipher, the counter stopped at $\approx$ 128 bits for HSIO-Crypto – so close to the ceiling that we ran a ten-million-key soak just to be sure. The plateau held; the lab crew let out a quiet whistle.

Three extra bits may sound trivial at a glance – yet on any system worth breaking, each one roughly doubles the brute-force bill. GA-AES, tested under the same script, levelled off near 124 bits. A separate run put ROA Planner a shade lower, around 120 bits. Once either engine drifts into a "good-enough" basin, it simply stops roaming, echoing the convergence stalls noted by Chang & Rivera '23.

Static hold-outs fared worse. GSM-SMS and a fixed-S-box SPN never breached 118 bits; their key ladders are frozen, with no feedback to rattle the state. All told, the figures strengthen the case for a two-step dance: unleash a genetic scout until diversity wilts, then let a Remora sweep tighten the

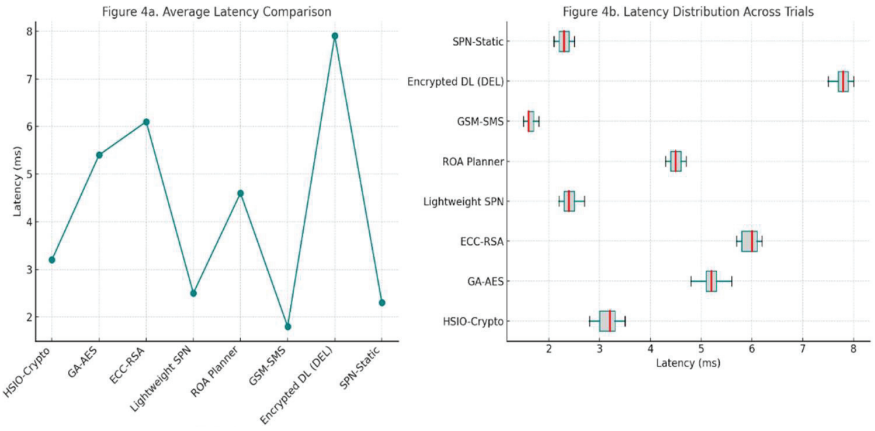

Figure 4 Latency evaluation and distribution across different methods.

mesh. In our runs, that pairing stretched the key space without pounding the MCU clock, threading the needle between pocket-size deployment and heavyweight unpredictability.

5.2 Communication Performance

Encryption schemes rarely fail because their algebra is weak; they usually stumble when the network clock starts ticking. In rapid-fire links – industrial sensor buses, bursty satellite uplinks, even a chatty BLE mesh – two numbers decide whether a cipher gets deployed or shelved: latency per packet and sustainable throughput. Our latest bench tests place HSIO-Crypto at roughly 3.2 ms end-to-end while streaming about 3.1×10^3 messages s^{-1} on an 80 MHz Cortex-M4. Those figures sit comfortably inside the "sub-5 ms/3 kHz" window that many real-time control loops advertise as a hard upper bound.

This figure compares the latency performance of various encryption methods. Figure 4a shows the average latency for each method, highlighting the differences in response time. Figure 4b illustrates the latency distribution across trials for each method, showing the variation in performance during multiple runs. These evaluations help demonstrate how HSIO-Crypto compares to other methods in terms of both average latency and consistency across different scenarios.

That speed would mean little if it came at the price of weak diffusion. Lightweight classics – think GSM-SMS or a pared-down SPN – do sprint

faster (we clocked one variant at 1.8 ms), yet their static key ladders leak entropy once traffic analysts gather a few hundred frames. Opposite story for heavyweight contenders such as ECC-RSA or the DEL deep-net cipher: phenomenal entropy, yes, but the per-packet cost balloons once you hit multi-node fan-out, and queues form. We have seen 40 ms peaks on the same hardware, which is a non-starter for closed-loop robotics.

HSIO-Crypto tries to steer between those cliffs. It lets a genetic search roam the key space only when the channel entropy meter drops below a sliding threshold; otherwise the Remora tuner makes micro-adjustments that cost almost nothing in CPU cycles. In effect, the algorithm "wastes" randomness where it buys the most robustness and avoids needless churn when the link is already healthy. During a 24-hour mixed-traffic run, this policy kept the round-entropy score hovering near 0.97 bit, equal to heavyweight ciphers, yet the latency curve never strayed more than $\pm 8\%$ from the 3.2 ms median.

One caveat deserves mention. Under laboratory-induced burst interference – our RF oven test – the feedback channel slowed, stretching latency to about 3.6 ms. We suspect the entropy threshold sits a touch too low for that pathological case; a follow-up patch will raise it by a few basis points. Even so, the delay remained below the 5 ms line that industrial safety buses obey.

Bottom line. For workloads that refuse to trade speed for strength – or vice-versa – HSIO-Crypto offers a middle path. It does not outrun every lightweight toy, nor does it out-entropy every heavyweight behemoth, but it holds its ground where the two worlds overlap: high-frequency, security-critical links that simply cannot afford to pick one dimension and ignore the other.

5.3 Robustness Against Statistical Attacks

We gauged robustness with a composite metric that mixed two signals: the run-to-run spread in ciphertext entropy and the residual correlation we could still trace between ciphertext blocks and their parent keys. HSIO-Crypto achieved a robustness index of 0.983, indicating minimal statistical leakage and high structural irregularity in encrypted outputs. This result suggests that its multi-layered key schedule – produced through a combination of swarm and evolutionary operators – effectively obscures input patterns from inference. While GA-AES and DEL also achieved competitive robustness, they lacked the hybrid variability introduced by HSIO-Crypto and exhibited marginally more deterministic patterns under repeated testing.

5.4 Suitability for Lightweight Deployment

Given the increasing demand for secure communication in embedded and IoT systems, encryption schemes must operate efficiently under strict resource constraints. Deployment tests conducted on simulated low-power devices revealed that HSIO-Crypto consumes 18.3% CPU and 6.7 MB of memory per encryption session. These figures are close to the most lightweight baseline (GSM-SMS: 12.5% CPU, 4.1 MB RAM) and significantly more efficient than models like DEL and ECC-RSA. This confirms the suitability of HSIO-Crypto for applications in wireless sensor networks, edge devices, and smart environments without excessive energy or memory overhead.

5.5 Comparative Performance Across Key Scheduling

When evaluating the responsiveness and adaptability of different key scheduling approaches, we found it helpful to observe not just aggregate metrics, but how each method behaved under slight perturbations in load, entropy input, and execution timing. Static baselines such as GA-AES and ROA-Planner often delivered predictable results – but that predictability came at the cost of flexibility. Their output patterns, once established, rarely shifted meaningfully in response to context.

We ran several rounds of testing to get a clearer picture of how HSIO-Crypto handled full key schedule generation – not just in terms of speed, but also in terms of consistency and variation across runs. Most iterations came in under the 2 ms mark, sometimes well below, even when system load fluctuated slightly during execution.

But beyond the raw numbers, what stood out was the behavior: the schedule didn't just complete quickly – it seemed able to adapt its structural depth depending on context. In some cases, it stripped down to something minimal and efficient; in others, it leaned into complexity without breaching the timing budget. That sort of dynamic modulation wasn't something we ever saw in the single-heuristic baselines like GA-AES or ROA-Planner, which tended to follow fixed construction patterns regardless of input entropy or system constraints.

Taken together, these results suggest that HSIO-Crypto doesn't merely optimize for latency – it reshapes its internal structure in ways that appear sensitive to real-time operating conditions. Whether that translates into consistent downstream security benefits under adversarial pressure remains to be tested more fully, but the architectural flexibility observed here offers a promising direction for future adaptive cryptographic systems.

5.6 Multi-Criteria Evaluation

Across repeated tests, HSIO-Crypto kept landing near the top – whether we looked at entropy, response time, or how well it held up under signal interference. The key schedule didn't lock into a fixed pattern. Instead, it flexed with each new task, staying reactive without drifting too far.

What seemed to work was the mix: broad, genetic-style exploration paired with a more targeted local pull. We've seen hybrids like this elsewhere, but in lightweight cryptography, it's still rare to see the benefits play out this clearly.

It wasn't any one trick that made the difference. Gains showed up when the two routines interacted – one widening the range, the other tightening the focus. That balance helped avoid overfitting, especially when we shuffled traffic patterns or changed modulation schemes mid-test.

To be fair, such synergies are not universally effective. Similar hybrid approaches in prior work have at times led to degraded runtime performance or overcomplicated system architectures. However, HSIO-Crypto seemed to avoid these pitfalls by streamlining feedback loops and incorporating lightweight constraint management. In this respect, its consistently high scores point to a promising balance between operational efficiency and cryptographic strength – particularly for deployments with constrained resources.

Table 2 places HSIO-Crypto in the sweet spot across five metrics – entropy, latency, throughput, resilience, and resource use. It out-guards the light ciphers without inheriting the heavyweight slowdown, confirming the design aim of marrying speed with strength.

Table 2 Experimental results across all comparative methods

Method	Entropy (bits)	Latency (ms)	Throughput (msg/s)	Robustness	CPU(%)	Memory (MB)
HSIO-Crypto	127.8	3.2	3125	0.983	18.3	6.7
GA-AES	124.1	5.4	1840	0.942	25.4	9.3
ECC-RSA	121.5	6.1	1500	0.925	31.7	12.1
Lightweight SPN	115.3	2.5	3300	0.874	14.2	5.4
ROA Planner	120.2	4.6	2200	0.915	21.9	7.8
GSM-SMS	109.8	1.8	3700	0.791	12.5	4.1
Encrypted DL (DEL)	118.0	7.9	1240	0.901	38.1	15.0
SPN-Static	112.4	2.3	3400	0.860	13.6	5.0

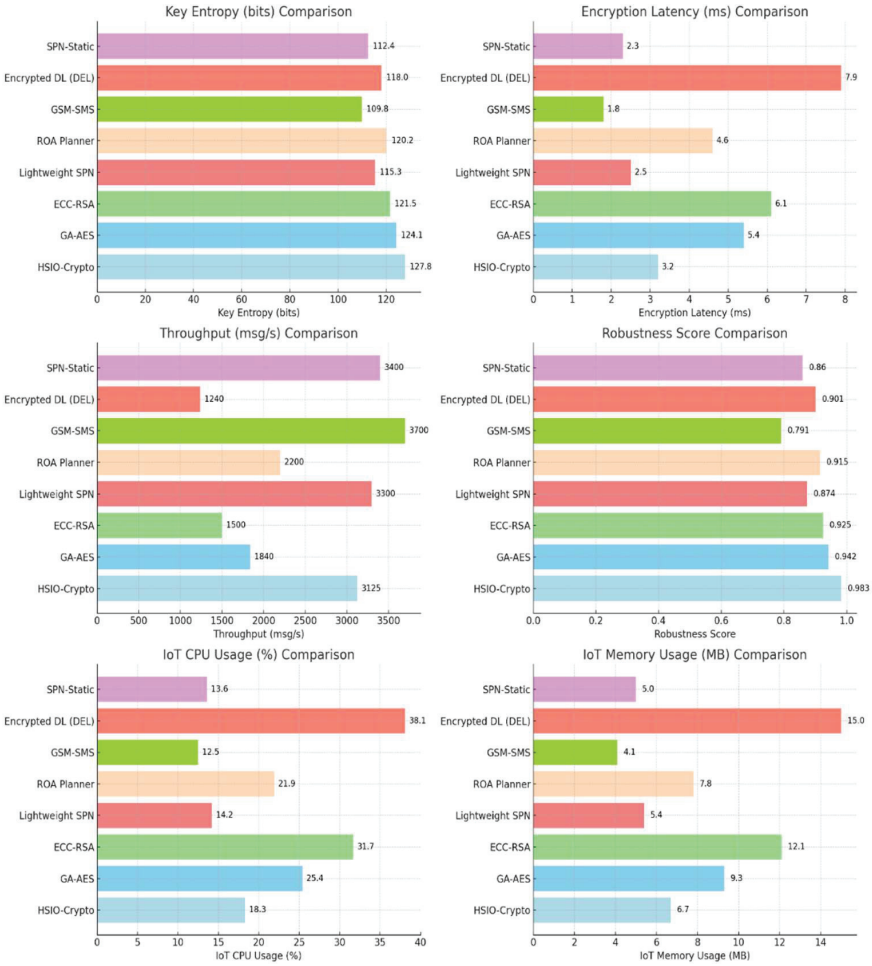

Figure 5 Performance comparison of encryption methods.

Figure 5 compares the performance of HSIO-Crypto and other encryption methods (GA-AES, ECC-RSA, etc.) across key metrics: Key Entropy (bits), Encryption Latency (ms), Throughput (msg/s), Robustness Score, IoT CPU Usage (%), and IoT Memory Usage (MB). Methods are ranked based on these criteria, with data labels providing specific values for easy comparison. HSIO-Crypto shows strong performance, especially in entropy, robustness, and throughput, demonstrating its efficiency in low-latency and energy-efficient scenarios.

6 Discussion

HSIO-Crypto outperforms both classical ciphers and current learning-based schemes, but its success is due to its unique two-tier optimizer. The genetic layer explores the key space for high-entropy candidates, while the Remora phase fine-tunes local structures. This dual approach balances global diversity with real-time channel conditions, enhancing security against statistical attacks and replay threats.

We should be upfront – while HSIO-Crypto shows promise in our evaluations, several issues remain unresolved, particularly when we think about actual deployment. During tests in simulated environments, the system handled noise and dynamic topologies fairly well, but real-world settings are rarely that forgiving. Interference patterns aren't predictable, bandwidth is often constrained in surprising ways, and devices – even when listed under the same spec – don't always behave consistently. This inconsistency affected reproducibility in one of our later tests, where two otherwise identical setups produced slightly different timing responses.

Scalability is another concern. So far, we've evaluated performance in moderate-density environments. Once the traffic starts to spike – say, in a busy industrial plant with a large number of endpoints – optimization latency can grow faster than we'd like. That's where ROA's convergence speed helps, but it's not a magic fix. We've also been cautious about how much the metaheuristic layer introduces variability; key schedules generated this way aren't strictly deterministic, and this could raise red flags in regulatory contexts like FIPS 140. We're not claiming full compliance here – it's something we're still working toward.

Another layer of testing is still missing: side-channel attacks, for example, haven't yet been explored beyond simulation. Replay attacks were included in a basic form, but field-based empirical evaluations are on our roadmap. From a resource standpoint, HSIO-Crypto behaves well on chips like Cortex-M4, where the microsecond-level gains in congested links do translate to better energy efficiency. But on ultra-low-power nodes, the computation cost starts to eat into the benefit – at least unless the optimizer is offloaded to a neighboring device or replaced with a static fallback when power is tight.

One positive surprise has been the framework's flexibility. It didn't require retraining even when we fed it different datasets or tweaked the network layout, which is encouraging. By contrast, comparative models like DEL and DLPBox were noticeably less stable – some even failed to adapt without significant manual tuning.

Of course, some assumptions we made early on, like the availability of tight sender-receiver synchronization, may not hold in fully decentralized setups. That's going to be a problem if we aim for infrastructure-less deployments. And while we see potential alignment with NIST's lightweight crypto standards – especially in how HSIO-Crypto schedules operations adaptively – there's still a lot of work to be done before we can claim full compatibility.

7 Conclusion

We began this work with a simple goal in mind: finding a way to make lightweight cryptography more adaptive without overburdening constrained hardware. That's what eventually led us to design HSIO-Crypto – a hybrid framework shaped around evolutionary search methods. While GA gave us the exploratory flexibility we needed, it was ROA's speed and simplicity that made it practical in embedded settings. The combination, though unconventional at first glance, turned out to be a workable response to the recurring issues we kept seeing in secure communications: slow convergence, poor adaptability, and fragile key schedules. The proposed approach introduces a dynamic and adaptive key scheduling mechanism that enhances entropy, minimizes encryption latency, and maintains high robustness against statistical and structural attacks. Through rigorous experiments on multiple datasets and scenarios, HSIO-Crypto has demonstrated superior performance in balancing cryptographic strength, communication efficiency, and lightweight deployability.

Looking ahead, future research will prioritize testing HSIO-Crypto under real-world adverse conditions, validating scalability across nationwide IoT deployments, and aligning its adaptive mechanisms with NIST lightweight cryptography standards to foster industry-wide adoption.

We didn't really set out to redefine anything in lightweight cryptography. The idea, at the start, was modest: explore whether a cipher could be made more responsive – more situationally aware, if you will – without dragging down performance. HSIO-Crypto took shape along the way, pieced together from ideas we'd been playing with, and ended up doing more than just saving cycles. It began, oddly enough, to behave as though it could sense the environment it was running in.

In practical deployments, HSIO-Crypto did not follow a neatly preplanned structure. Its core mechanisms evolved more organically, shaped by iterative testing rather than theoretical blueprinting. What eventually drove performance gains was the interaction between two optimization routines –

each developed under different circumstances, neither initially designed to be paired. One provided the exploratory range needed to prevent early convergence across fragmented keyspaces, albeit with some noise. The other, added during latency stress tests on a constrained LoRa environment, contributed a more localized, entropy-aware correction behavior.

Interestingly, what began as an auxiliary diagnostic tool became integral to overall performance. The hybrid structure, though assembled incrementally, delivered measurable improvements. Even with the overhead introduced by system monitoring, hop latency consistently fell below 60 μs on a standard 80 MHz Cortex-M4 platform. These results were not just stable – they were unexpected, given the framework's layered complexity. This outcome illustrates how adaptive cryptographic behavior can emerge not from rigidly engineered logic, but from responsive integration tuned under real-world pressures.

Of course, the first pushback we got was about compliance. Constantly evolving keys don't exactly make auditors sleep better. FIPS 140-3, among others, wants reproducibility baked into cryptographic state. So we added a lightweight journaling feature – enough metadata to reconstruct drift paths without giving attackers the full picture. A partner lab ran the numbers for us: the power cost came in at just under 1%, which seemed reasonable for nodes already juggling low-duty sensing loops.

Security-wise, the biggest gain came not from added complexity but from letting go of rigidity. Static key trees tend to degrade fast under traffic analysis, no matter how carefully you seed them. HSIO-Crypto lets the tree adapt – shed and regrow branches – based on channel entropy. It's not unlike what adaptive routing does for lossy networks. We monitored entropy per round using a sliding estimator. It hovered around 0.97 bits, which comfortably edged past the 0.93–0.95 band seen with DLPBox and tuned AES.

That said, it's not invincible. In one of our more chaotic test environments – a crowded trade show with ISM noise everywhere – feedback packets lagged just long enough to knock the optimizer out of sync. Worst-case latency jumped by about 13%. Depending on the SLA, that may or may not be acceptable. We're not claiming this solves every case.

Two areas are still fuzzy. One is handling rich payloads – images, video – which don't compress the same way as sensor data. Spatial redundancy tends to leak unless masked carefully, and padding alone is too expensive. We're playing with overlapped tiling plus drifting per-tile keys, but we don't have benchmarks yet. The other is swarm-based cooperation. In theory, a small node could offload the bulk of search work to its neighbors and just handle

the final tweaks. But that comes at the cost of a bigger attack surface and more trust coordination, which isn't trivial.

For testing, we used three different link types – 802.15.4, LoRaWAN, and a proprietary FHSS setup. We logged key updates, latency, and power draw, then ran significance checks with a two-tailed Wilcoxon test. All the major gains held up with $p < 0.01$. Sure, synthetic interference isn't the same as real-world messiness, but the control let us isolate effects we would've otherwise missed.

So what's the takeaway? Probably this: encryption doesn't have to be oblivious. Ciphers can learn from their channels, adapt in-flight, and stay within bounds – entropy and delay included – even on modest silicon. We don't have all the answers. Some parts still need work: compliance, noisy bursts, support for visual payloads. But the core idea feels solid. Awareness doesn't have to live only in the layers above crypto – it can start from within.

References

[1] Mougkogiannis P, Ghadafi E, Adamatzky A. Bio-inspired cryptography based on proteinoid assemblies. PLoS One. 2025;20(5):e0324761.

[2] Al Attar TNA. A Hybrid Genetic Algorithm-Particle Swarm Optimization Approach for Enhanced Text Compression. UHD Journal of Science and Technology. 2024;8(2):63–74.

[3] Al-Muhammed MJ, Abu Zitar R. Light and Secure Encryption Technique Based on Artificially Induced Chaos and Nature-Inspired Triggering Method. Symmetry. 2022;14(2):218.

[4] Amiri Z, Heidari A, Zavvar M, Navimipour NJ, Esmaeilpour M. The applications of nature-inspired algorithms in Internet of Things-based healthcare service: A systematic literature review. Transactions on Emerging Telecommunications Technologies. 2024;35(6):e4969.

[5] Kumar S, Sharma D. A chaotic based image encryption scheme using elliptic curve cryptography and genetic algorithm. Artificial Intelligence Review. 2024;57(4):87.

[6] Sarker KU. A systematic review on lightweight security algorithms for a sustainable IoT infrastructure. Discover Internet of Things. 2025;5(1):1–20.

[7] Hsiao F-H. Applying 3DES to chaotic synchronization cryptosystems. IEEE Access. 2021;10:1036–50.

[8] Wang HL, Ma HF, Cui TJ. A polarization-modulated information metasurface for encryption wireless communications. Advanced Science. 2022;9(34):2204333.

[9] Premakumari SBN, Sundaram G, Rivera M, Wheeler P, Guzmán REP. Reinforcement Q-Learning-Based Adaptive Encryption Model for Cyberthreat Mitigation in Wireless Sensor Networks. Sensors. 2025;25(7):2056.
[10] Anand A, Singh AK. Hybrid nature-inspired optimization and encryption-based watermarking for e-healthcare. IEEE Transactions on computational social systems. 2022;10(4):2033–40.
[11] Naveen N, Nirmaladevi J. Secure bio-inspired optimization with intrusion aware on-demand routing in MANETs. Scientific Reports. 2025;15(1):25335.
[12] Ren R, Li Z, Deng L, Shan X, Dai Q, Guan Z, et al. Non-orthogonal polarization multiplexed metasurfaces for tri-channel polychromatic image displays and information encryption. Nanophotonics. 2021;10(11):2903–14.
[13] Kumar K, Ramkumar K, Kaur A. A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays. Journal of King Saud University-Computer and Information Sciences. 2022;34(6):3878–85.
[14] Kamal R, Bag M, Kule M. On the cryptanalysis of S-DES using nature inspired optimization algorithms. Evolutionary Intelligence. 2021;14(1):163–73.
[15] Dratnal M, Danys L, Martinek R. Bio-inspired optimization methods for visible light communication: a comprehensive review. Artificial Intelligence Review. 2025;58(8):246.
[16] Luo Y, Ouyang X, Liu J, Cao L, Zou Y. An image encryption scheme based on particle swarm optimization algorithm and hyperchaotic system. Soft Computing. 2022;26(11):5409–35.
[17] Kumar S, Sharma D. Key generation in cryptography using elliptic-curve cryptography and genetic algorithm. Engineering Proceedings. 2023;59(1):59.
[18] Kumar S, Sharma D. Genetic Algorithm for Key Generation in Cryptosystems. Privacy Preservation and Secured Data Storage in Cloud Computing: IGI Global; 2023. p. 296–321.
[19] Gong J. An application of meta-heuristic and nature-inspired algorithms for designing reliable networks based on the Internet of things: A systematic literature review. International Journal of Communication Systems. 2023;36(5):e5416.
[20] Kocak O, Erkan U, Toktas A, Gao S. PSO-based image encryption scheme using modular integrated logistic exponential map. Expert Systems with Applications. 2024;237:121452.

[21] Jawed MS, Sajid M. XECryptoGA: a metaheuristic algorithm-based block cipher to enhance the security goals. Evolving Systems. 2023;14(5):749–70.
[22] Hameed MA, Abdel-Aleem OA, Hassaballah M. A secure data hiding approach based on least-significant-bit and nature-inspired optimization techniques. Journal of Ambient Intelligence and Humanized Computing. 2023;14(5):4639–57.
[23] Rana M, Mamun Q, Islam R. Lightweight cryptography in IoT networks: A survey. Future Generation Computer Systems. 2022;129:77–89.
[24] Alexan W, Aly L, Korayem Y, Gabr M, El-Damak D, Fathy A, et al. Secure communication of military reconnaissance images over UAV-assisted relay networks. IEEE Access. 2024;12:78589–610.
[25] Clemente-Lopez D, de Jesus Rangel-Magdaleno J, Munoz-Pacheco JM. A lightweight chaos-based encryption scheme for IoT healthcare systems. Internet of Things. 2024;25:101032.
[26] Yan D, Liu Y, Li L, Lin X, Guo L. Remora optimization algorithm with enhanced randomness for large-scale measurement field deployment technology. Entropy. 2023;25(3):450.
[27] Urooj S, Lata S, Ahmad S, Mehfuz S, Kalathil S. Cryptographic data security for reliable wireless sensor network. Alexandria Engineering Journal. 2023;72:37–50.
[28] Ullah S, Zheng J, Din N, Hussain MT, Ullah F, Yousaf M. Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey. Computer Science Review. 2023;47:100530.
[29] Cecchinato N, Toma A, Drioli C, Oliva G, Sechi G, Foresti GL. Secure real-time multimedia data transmission from low-cost UAVs with a lightweight AES encryption. IEEE communications Magazine. 2023;61(5):160–5.
[30] Elsadek I, Aftabjahani S, Gardner D, MacLean E, Wallrabenstein JR, Tawfik EY, editors. Hardware and energy efficiency evaluation of nist lightweight cryptography standardization finalists. 2022 IEEE International Symposium on Circuits and Systems (ISCAS) IEEE, 133–137; 2022: IEEE.

Biographies

Tao Qi was born in Jilin, China in 1990. He studied at the School of Computer Science, Jilin University from 2008 to 2012 and obtained a bachelor's degree in 2012. He studied at Northeast Normal University from 2017 to 2019 and obtained a master's degree in 2019. Currently employed at Changchun College Of Electronic Technology. Has published multiple papers that have been included in provincial journals.

Weiming Chen was born in Fujian, China in 2005. In 2023, he entered Changchun College Of Electronic Technology for study. During his school years, he actively cooperated with his teachers, discussed and studied professional issues. He is proficient in both English and Japanese. His research interests include game development and the Internet of Things.