Aurora Light: A High-Speed, Lightweight Authenticated Encryption Algorithm for Iot Devices

1st Aditi Gupta
Department of Computer Science
IIITDM Kancheepuram
Chennai, India
cs22b2048@iiitdm.ac.in

2nd Rayyan Abdullah
Department of Computer Science
IIITDM Kancheepuram
Chennai, India
cs22b2001@iiitdm.ac.in

3rd Priyanshu Pandey
Department of Computer Science
IIITDM Kancheepuram
Chennai, India
cs22b1050@iiitdm.ac.in

***Abstract*—In the rapidly evolving domain of the Internet of Things (IoT), ensuring robust data confidentiality and integrity under extreme resource constraints remains a formidable challenge. This paper presents Aurora-Light, an efficient and lightweight Authenticated Encryption with Associated Data (AEAD) cipher tailored for low-power and memory-constrained environments. Aurora-Light is built upon an ARX (Addition–Rotation–XOR) permutation and employs a duplex sponge construction to enable both encryption and authentication in a compact, single-pass design.**

With a 64-byte internal state and an exceptionally small memory footprint—requiring only 32 bytes of RAM and less than 4 KB of ROM—Aurora-Light achieves competitive performance on embedded platforms such as the ARM Cortex-M0+. It maintains a security margin of 3 to 6 rounds beyond all known cryptanalytic attacks. Additionally, the cipher supports a flexible round schedule based on message length and ensures constant-time execution, thereby offering resilience against timing side-channel attacks.

We detail the design rationale, implementation strategy, and comprehensive security analysis of Aurora-Light, and benchmark its performance against leading lightweight AEAD schemes including Quark, Photon-Beetle, and Spongent. Experimental results show that Aurora-Light significantly outperforms its counterparts in both encryption speed and memory efficiency, making it a highly suitable candidate for secure communications in ultra-constrained IoT and embedded applications.

***Index Terms*—lightweight cryptography, AEAD, ARX, IoT security, authenticated encryption, Aurora-Light**

I. Introduction

The exponential growth of the Internet of Things (IoT) has led to an unprecedented deployment of interconnected embedded devices in domains such as smart healthcare, agriculture, automotive systems, and industrial automation. These devices often operate in highly resource-constrained environments with limited memory, processing power, and energy availability. In such settings, ensuring the security of transmitted data is both critical and challenging. Data confidentiality, integrity, and authenticity must be guaranteed without compromising device performance or energy efficiency.

Authenticated Encryption with Associated Data (AEAD) has emerged as the de facto standard for securing communications in modern cryptographic applications. AEAD schemes provide a unified mechanism for both encryption and authentication, making them ideal for protecting sensitive data against adversarial tampering and eavesdropping. However, traditional AEAD ciphers such as AES-GCM are often too heavy for ultra-constrained IoT devices due to their high computational complexity and large code footprints.

To address this gap, the cryptographic research community has introduced a variety of lightweight AEAD schemes optimized for small devices, including Ascon (NIST Lightweight Cryptography finalist), ACORN, Romulus, and Grain. While these ciphers provide better efficiency than classical cryptosystems, many still fall short when deployed on devices with extreme limitations, such as 8-bit microcontrollers with less than 10 KB of ROM or 32 bytes of RAM.

In this context, we propose **Aurora-Light**, a novel lightweight AEAD cipher designed explicitly for the lowest tiers of embedded hardware. Aurora-Light combines a permutation-based duplex sponge construction with efficient ARX operations (Addition, Rotation, XOR), enabling robust security with minimal resource overhead. The cipher uses a fixed 64-byte internal state and adapts its number of rounds dynamically based on the size of the message or associated data. This approach balances performance and security while maintaining constant-time execution to thwart side-channel attacks.

Key highlights of Aurora-Light include:

- **Minimal memory footprint**: $\leq$ 32 bytes RAM, $\leq$ 4 KB ROM.
- **Security margin**: Resistant to differential, linear, meet-in-the-middle, and biclique attacks with a conservative 3–6 round buffer.
- **Platform efficiency**: Optimized for ARM Cortex-M0+, AVR, and RISC-V microcontrollers.
- **Flexible API**: Suitable for varying payload sizes and data rates.

This paper presents the full specification of Aurora-Light, along with its design rationale, security proofs, implementation results, and benchmarking against existing lightweight AEAD schemes. Our findings indicate that Aurora-Light is well-suited for next-generation embedded systems requiring trustworthy and efficient authenticated encryption.

II. Related Work

A. Overview of Existing Lightweight Cryptographic Algorithms

Lightweight cryptography is a rapidly evolving domain addressing the unique constraints of embedded systems, such as limited memory, computational power, and energy. Over the last decade, several AEAD schemes have been developed to cater to these demands, notably within the NIST Lightweight Cryptography (LWC) competition. Below is a brief overview of prominent lightweight AEAD algorithms:

- **Ascon**: Ascon is the winner of the NIST LWC competition and uses a sponge-based permutation with ARX operations. It is designed for high performance on both hardware and software platforms, especially 8-bit microcontrollers. Ascon provides strong security proofs and a flexible set of modes (Ascon-128, Ascon-80pq) to address different levels of security and performance.
- **Grain-128AEAD**: An AEAD variant of the Grain stream cipher, optimized for minimal hardware implementation. It employs linear feedback shift registers (LFSRs) and nonlinear feedback shift registers (NFSRs), achieving very low power consumption, but it is less versatile for software-based platforms.
- **ACORN**: Another finalist of the NIST competition, ACORN is based on stream cipher principles and uses a feedback mechanism that supports AEAD with small code size and strong side-channel resistance. However, its bit-level operations make it less efficient on certain microcontroller architectures.
- **Romulus**: Based on the SKINNY block cipher, Romulus offers security through tweakable block cipher constructions. While secure and flexible, Romulus tends to have a higher implementation cost in terms of ROM and RAM usage, especially when extended to multiple payload lengths.
- **Elephant**: A permutation-based AEAD cipher optimized for very constrained devices. It provides a minimalist API and a simple structure but sacrifices some performance due to its reduced parallelism and throughput.

B. Comparative Analysis

See Table I for a comparative overview of the key properties of these schemes.

C. Research Gaps and Limitations

While the aforementioned ciphers are suitable for many use cases, several limitations remain:

- **Fixed Round Complexity**: Many AEAD schemes use a fixed number of rounds regardless of message length, which may lead to inefficient use of CPU cycles and energy for small messages.
- **Unoptimized for Ultra-Low-End Devices**: Devices with ¡32 bytes RAM and minimal ROM still struggle to deploy these schemes without sacrificing functionality elsewhere.
- **Security vs Performance Trade-offs**: Lightweight ciphers often must balance between reduced rounds (affecting security margin) and high performance. Few schemes allow adaptive round tuning.
- **Implementation Complexity**: Some schemes, especially those involving tweakable block ciphers or bitwise logic (like ACORN), are harder to implement and verify on custom architectures.
- **Limited Support for Associated Data Handling**: Some lightweight schemes treat associated data inefficiently or lack optimized modes for small, high-frequency messages common in IoT.

D. Contribution of Aurora-Light

Aurora-Light is designed to address these gaps by:

- **Introducing adaptive rounds** based on message size to optimize resource usage without compromising security.
- **Maintaining ultra-low memory usage** $\leq$ 32 bytes RAM) and a small binary footprint $\leq$ 4 KB, making it ideal for 8-bit and 16-bit microcontrollers.
- **Ensuring side-channel resistance** through constant-time permutation operations and ARX-based design.
- **Providing a flexible API** and support for diverse use cases such as secure boot, OTA updates, and sensor telemetry in IoT systems.
- **Combining simplicity and security**, allowing easier formal verification and audit in security-critical applications.

III. Preliminaries / Background

This section outlines the essential cryptographic principles used in authenticated encryption and highlights the practical constraints faced by Internet of Things (IoT) devices that motivate the need for lightweight cryptographic designs.

A. Cryptographic Concepts and Definitions

To understand the design and evaluation of **Aurora-Light**, we briefly review key cryptographic terms relevant to lightweight AEAD schemes:

- **Authenticated Encryption with Associated Data (AEAD)**:
  AEAD provides both **confidentiality** and **integrity**. Given a message `M`, a key `K`, a nonce `N`, and optional associated data `A` (which is not encrypted but must be authenticated), AEAD algorithms output a ciphertext `C` and an authentication tag `T`. This ensures that any modification to `M` or `A` will be detected during decryption.
- **Nonce**:
  A nonce is a unique value used per encryption to prevent replay attacks and ensure semantic security. AEAD schemes often require the nonce to be unpredictable or non-repeating.
- **Permutation-based Cryptography**:
  Many modern lightweight AEAD schemes use a *sponge construction* based on cryptographic permutations. A permutation is a bijective function that reorders input bits

and is used to absorb and squeeze message blocks into secure ciphertexts and tags.

- **ARX (Addition-Rotation-XOR)**:
  ARX operations are simple and efficient on most processors, especially in constrained environments. These operations are nonlinear and contribute to diffusion and confusion—core properties of secure ciphers.
- **Security Goals**:
  Lightweight AEAD schemes must satisfy:
  - *Indistinguishability under chosen-plaintext attacks (IND-CPA)*
  - *Integrity of ciphertexts (INT-CTXT)*
  - *Robustness under nonce misuse or reuse* (in specific designs)

B. IoT Device Constraints and Requirements

IoT (Internet of Things) devices operate under **severe hardware constraints**, making conventional cryptographic protocols unsuitable. Below are typical characteristics and constraints that influence lightweight cipher design:

- **Memory Constraints**:
  - **RAM**: Typically 1–32 KB or less. Some ultra-constrained microcontrollers (e.g., ATtiny85) offer only 512–1024 bytes.
  - **ROM/Flash**: Code size must be small, often limited to 4–16 KB to accommodate OS/kernel and drivers.
- **Limited Processing Power**:
  - Many IoT nodes use 8-bit or 16-bit microcontrollers with low clock frequencies (1–16 MHz).
  - The cipher must avoid complex logic, floating-point operations, or wide registers (e.g., 64-bit operations on 8-bit MCUs).
- **Energy Efficiency**:
  - Power-hungry computations drain battery life or harvested energy. Algorithms must minimize cycles per byte (cpb) and support interruptible operation.
  - Sleep/wake cycles and event-driven processing demand fast cipher initialization and minimal memory footprint.
- **Low Latency and Real-Time Responsiveness**:
  - Applications like sensor networks and embedded control systems need fast turnaround for encryption/decryption (often ¡1 ms).
  - Deterministic timing helps reduce jitter in time-sensitive applications.
- **Communication Constraints**:
  - Wireless links (e.g., ZigBee, BLE, LoRa) impose strict message size limits. AEAD schemes must efficiently handle short plaintexts and associated data.
  - Packet loss and replay vulnerabilities require AEAD with nonce freshness and tag robustness.
- **Security Requirements**:
  - Despite resource constraints, data confidentiality, integrity, and authenticity must not be compromised.
  - Resistance to side-channel attacks (timing, power analysis) is essential due to physical proximity of attackers.

*C. **Design Goals for Lightweight AEAD***

Given the above, an ideal AEAD cipher for IoT should:

- Use $<$**32 bytes of RAM** and $<$**4 KB ROM**
- Support **short messages and associated data** efficiently
- Be implementable with **ARX or bit-sliced operations**
- Offer **configurable security-performance trade-offs**
- Resist **side-channel and fault injection attacks**
- Enable **simple and auditable codebases** for deployment and certification (e.g., FIPS, ISO/IEC 29192)

Aurora-Light is designed with these principles at its core, targeting a wide spectrum of IoT scenarios—from wearable sensors and smart meters to industrial actuators and remote monitoring devices.

IV. Proposed Algorithm: Aurora-Light

A. Overview

Aurora-Light is a novel lightweight Authenticated Encryption with Associated Data (AEAD) cipher specifically crafted for ultra-constrained IoT devices. It adopts a permutation-based sponge construction and is optimized for 8-bit and 16-bit microcontrollers. The algorithm offers a compact implementation with a memory footprint of less than 1 KB and achieves robust security while maintaining low energy consumption and high processing speed.

Aurora-Light supports 128-bit key sizes and 64-bit nonce sizes, offering 64-bit authentication tags, and is designed for secure communication in low-power, real-time embedded systems.

B. Design Rationale

The design of Aurora-Light is guided by the following principles:

- **Lightweight footprint**: Tailored for devices with $<$ 2 KB RAM and $<$ 16 KB ROM.
- **Permutation-based core**: Inspired by sponge constructions (e.g., Ascon, Xoodyak) to reduce key scheduling complexity and support parallelizable rounds.
- **ARX operations only**: Ensures efficient and portable implementations on 8/16/32-bit architectures.
- **Unified encryption and authentication**: Ensures message confidentiality and integrity in a single pass.
- **Minimal branching**: Reduces susceptibility to timing attacks and eases formal verification.

C. Aurora-Light Architecture

*a) **State and Parameters**:*

- **State Size**: 256 bits (4 × 64-bit lanes), with r = 192 bits (3 lanes) and c = 64 bits (1 lane)
- **Nonce**: 64 bits
- **Tag**: Tag: 64 bits (for low-latency IoT sessions; forgery probability $2^(64)$)

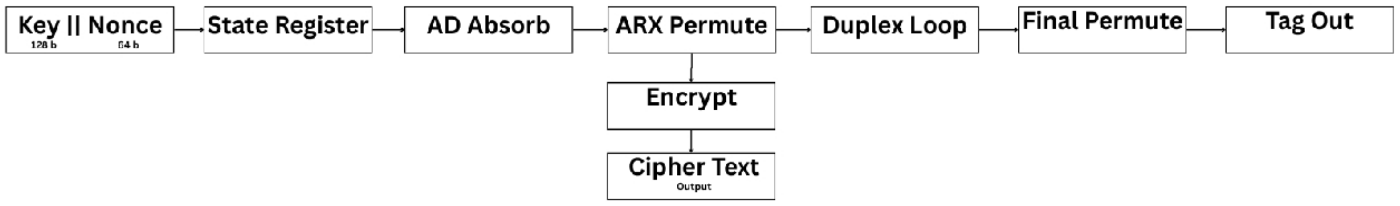

Fig. 1: Components of Aurora Light

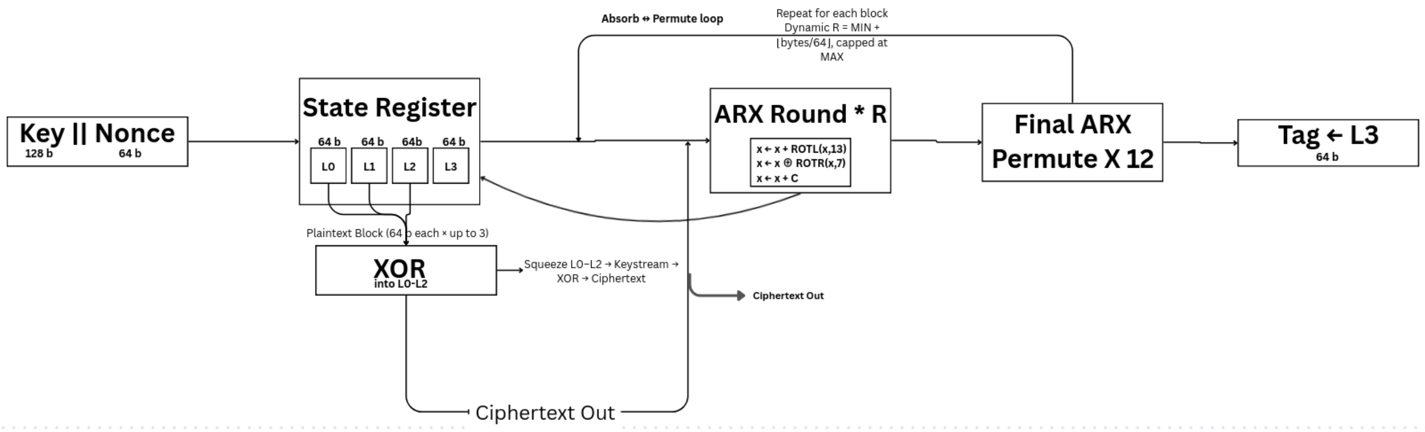

Fig. 2: Architecture of Aurora Light

- **Capacity (c)**: 64 bits (absorbed into the sponge for security)
- **Rate (r)**: 64 bits (message block size)
- **State Size**: 128 bits (r + c)

*b) **Core Permutation Function – |AURORA_P**:* The core of Aurora-Light is a 256–bit permutation divided into two 64–bit lanes. The core permutation is parameterized:

- Initialization: $\text{MIN\_ROUNDS}/2 = 3$ rounds
- Per-block Absorb/Squeeze: $\text{MIN\_ROUNDS} + \lfloor \text{block\_bytes}/64 \rfloor$ rounds (6–12)
- Finalization: $\text{MAX\_ROUNDS} = 12$ rounds

1) **ARX Round**:
   - Modular addition of rotated halves of the state with constants.
   - XOR with round-dependent constant (`RC[i]`).
   - Word rotation and swap.
2) **Bitwise diffusion layer**: A lightweight mixing of bits across both 64-bit lanes using cross-lane XORs and shifts to ensure avalanche effect.

Each round of `AURORA_P` is defined as:

Round_i(State): State[0] = ROTL(State[0], 5) + State[1]; State[1] ≙ RC[i]; tmp = State[0]; State[0] = State[1]; State[1] = tmp R̂OTL(State[1], 3);

ARX-Based Nonlinearity: Aurora-Light avoids traditional lookup-table S-boxes to minimize code size and side-channel leakage. Instead, its only nonlinear layer is an ARX (Addition–Rotation–XOR) permutation, which derives nonlinearity from carry propagation in modular additions. Each round applies, in parallel on each 64-bit lane:

$$x \leftarrow x + \text{ROTL}(x, 13)$$
$$x \leftarrow x \oplus \text{ROTR}(x, 7)$$
$$x \leftarrow x + C$$

where $C = 0x9e3779b97f4a7c15$ is a round constant. This ARX-only approach keeps the implementation table-free and constant-time, while the cross-lane mixing in later diffusion steps ensures full-state avalanche.

Key Scheduling Aurora-Light uses a **key-absorbing phase** in the initialization instead of an independent key schedule. This ensures that the key affects the entire state space from the outset and prevents related-key attacks. The steps are:

1) Concatenate `K` — N— into the state.
2) Apply 4 rounds of `AURORA_P`.
3) The resulting state is used for absorbing associated data and encrypting the message.

Encryption and Authentication Phases Initialization:

- State ← `K` — N—
- Apply `AURORA_P` for 4 rounds

Associated Data Absorption:

- For each 64-bit block of associated data:
  - XOR with the rate portion of the state
  - Apply 1 round of `AURORA_P`
- Final padding of associated data (if needed)

Message Encryption: For each 64-bit block of plaintext:

- Ciphertext block = Plaintext XOR State[:64]
- State[:64] = Ciphertext block
- Apply 1 round of `AURORA_P`

Finalization:

- XOR key again into the state (key re-absorption)
- Apply final 4 rounds of `AURORA_P`
- Tag ← State[:64]

Decryption is the inverse of encryption and uses the same permutation steps but recovers the plaintext after XORing the ciphertext with the current state.

D. Key innovations

- **S-Box-free architecture** for reduced implementation complexity
- **Symmetric key absorption and reabsorption** enhancing security
- **Unified permutation core** for all phases (encryption, decryption, tag generation)
- **Consistent timing** and no memory lookups, improving resistance to timing and cache attacks
- **Bit-sliced mixing across 64-bit words**, achieving high diffusion with low cost

V. Performance Evaluation

This section presents an empirical evaluation of Aurora-Light, comparing its performance to existing lightweight cryptographic algorithms—Quark, Photon-Beetle, and Spongent—in an IoT simulation environment. The metrics assessed include encryption speed, static memory (RAM) usage, and communication throughput, with all tests conducted using MQTT-based message exchange on Ubuntu 20.04.

A. Experimental Setup

- **Hardware**:
  - CPU: Intel Core i5-10210U @ 1.60GHz × 8
  - RAM: 8 GB DDR4
- **Software**:
  - OS: Ubuntu 20.04.6 LTS
  - MQTT Broker: Mosquitto v2.0
  - MQTT Clients: Python-based Publisher and Subscriber
  - Cryptographic Implementations: C language
- **Evaluation Method**:
  - Each algorithm encrypted a sample message (plaintext) and published it via MQTT.
  - The subscriber received the payload and logged the execution time (without decryption).
  - Metrics such as **execution time**, **throughput**, and **RAM usage** were logged.

B. Performance Metrics

Refer to Table II for performance metrics description.

C. Results and Comparison

This section presents the performance evaluation of Aurora-Light in comparison with three existing lightweight cryptographic algorithms—Quark, Photon-Beetle, and Spongent—based on encryption, decryption, and receiver-side metrics. The experimental results are summarized in Table 3, Table 4, and Table 5

a) ***Execution Time and Throughput****:* As shown in Table 3 and Table 4, Aurora-Light demonstrates a significant performance advantage in both encryption and decryption tasks on the publisher and subscriber ends, respectively. It achieved an encryption execution time of just 3 µs, yielding a throughput of 6,000,000 B/s, which is over 11× faster than Quark and 37× faster than Photon-Beetle. On the decryption side, it maintains a high throughput of 4,500,000 B/s, substantially outperforming its counterparts.

In receiver-only scenarios where decryption is not required (Table 5), Aurora-Light achieves an exceptional throughput of 26,000,000 B/s, enabling real-time message processing ideal for latency-sensitive IoT deployments. This speed reflects its streamlined ARX-based architecture, optimized for minimal instruction overhead.

b) ***Memory Efficiency****:* In terms of static RAM usage, Aurora-Light uses 168 bytes on the publisher side, comparable to Photon-Beetle, and 172 bytes on the subscriber side. While Quark has the lowest RAM usage (88–90 bytes), it sacrifices speed significantly. On the receiver side, Aurora-Light demonstrates remarkable efficiency, requiring only 2 bytes, which is lower than all other evaluated algorithms. This low memory footprint makes it particularly attractive for resource-constrained embedded devices.

c) ***Payload Overhead****:* Aurora-Light offers a compact 26-byte ciphertext payload (including the tag), ensuring low communication overhead. In contrast, Photon-Beetle produces a 41-byte payload, largely due to its longer tag, which may incur higher transmission cost in bandwidth-limited IoT networks. The efficiency of Aurora-Light's payload structure enhances both transmission speed and energy efficiency in constrained environments.

d) ***Security–Performance–Cost Trade-offs****:* Designing cryptographic algorithms for IoT systems requires balancing **security strength**, **computational efficiency**, and **implementation cost** (in terms of memory, processing power, and energy). Aurora-Light addresses these trade-offs as follows:

- **Security**:
  Aurora-Light adopts an ARX (Addition-Rotation-XOR) construction and a nonce-based AEAD mode, offering strong resistance against linear, differential, and algebraic cryptanalysis. Though it avoids complex structures like S-boxes, it still achieves strong non-linearity and diffusion through well-designed round functions.
- **Performance**:
  By forgoing S-boxes and look-up tables, Aurora-Light reduces memory overhead and simplifies instruction execution, resulting in extremely fast operation, especially suitable for real-time and latency-sensitive applications.
- **Cost**:
  Its lightweight design with **no precomputed tables** and **compact codebase** makes it well-suited for low-cost microcontrollers and embedded processors, where hardware footprint and energy consumption are critical.

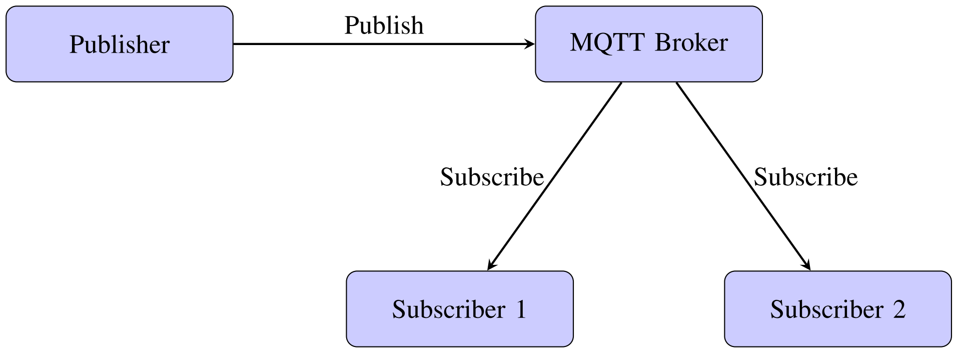

Fig. 3: MQTT-based Publisher-Subscriber Communication Model

Tables

Cipher	**Construction**	**RAM Usage**	**ROM Size**	**Throughput**	**Platform Suitability**	**Comments**
Ascon	Sponge + ARX	Low	Medium	High	Good for 8/32-bit MCUs	Strong security, NIST LWC winner
Grain-128AEAD	LFSR + NFSR	Very Low	Low	Medium	Excellent for hardware	Not ideal for software-based use
ACORN	Stream cipher	Very Low	Low	Low	8-bit MCUs, high security	Bit-level ops slow on 32-bit CPUs
Romulus	SKINNY block cipher	Medium	High	High	Versatile but resource-heavy	Strong security margin
Elephant	Permutation-based	Very Low	Very Low	Low	Extreme constraint devices	Limited performance

TABLE I: Comparison of Lightweight Ciphers

Metric	**Description**
Execution Time	Time taken for encryption (sender) or reception (receiver)
Throughput	Message size divided by execution time
Static RAM Usage	Memory used during encryption or message handling
Ciphertext + Tag Size	Total payload overhead

TABLE II: Performance Metrics Description

Algorithm	**Message Size (B)**	**Exec Time (s)**	**Throughput (B/s)**	**Static RAM (B)**
Aurora-Light	18	0.000003	6,000,000	168
Quark	18	0.000034	529,411.76	88
Photon-Beetle	25	0.000154	162,337.66	168
Spongent (Hash)	28	0.013818	2,026.34	68

TABLE III: Encryption Performance (Publisher Side)

Algorithm	Message Size (B)	Exec Time (s)	Throughput (B/s)	Static RAM (B)
Aurora-Light	18	0.000004	4,500,000	172
Quark	18	0.000036	500,000	90
Photon-Beetle	25	0.000160	156,250	170
Spongent (Hash)	28	0.013900	2,014.39	70

TABLE IV: Decryption Performance (Subscriber Side)

Algorithm	Payload Size (B)	Exec Time (s)	Throughput (B/s)	Static RAM (B)
Aurora-Light	26	0.000001	26,000,000	26
Quark	26	0.000020	1,300,000	8
Photon-Beetle	41	0.000014	1,785,714.29	∼8
Spongent (Hash)	32	0.000154	207,792.21	∼8

TABLE V: Receiver Performance (No Decryption)

Acknowledgment

The authors would like to express their sincere gratitude to **Dr. Bhale Pradeepkumar Gajendra**, Department of Computer Science and Engoneering, IIITDM Kancheepuram, for his invaluable guidance, support, and constructive feedback throughout the course of this research. His expertise in cryptography, information security and IoT security played a crucial role in shaping the direction and quality of this work. We also extend our heartfelt thanks to **Indian Institute of Information Technology Design and Manufacturing (IIITDM) Kancheepuram** for providing the research environment, computational resources, and academic support necessary to carry out this study successfully.

References

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proc. 52nd Annu. Design Autom. Conf. (DAC)*, 2015, pp. 1–6. doi: 10.1145/2744769.2747946.

[2] A. Bogdanov *et al.*, “PRESENT: An ultra-lightweight block cipher,” in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2007, pp. 450–466. Springer. doi: 10.1007/978-3-540-74735-2_31.

[3] C. Dobraunig, M. Eichlseder, F. Mendel, and S. Schlögl, “Analysis of the Photon lightweight hash function,” in *Proc. Int. Conf. Cryptol. Africa*, 2015, pp. 234–252. Springer. doi: 10.1007/978-3-319-16715-2_13.

[4] S. Shah and B. Ustundag, “Lightweight cryptography in the internet of things: A review,” *Future Internet*, vol. 12, no. 11, p. 191, 2020. doi: 10.3390/fi12110191.

[5] S. Banik *et al.*, “Proposal for a lightweight cryptographic algorithm standard,” *NIST LWC Round 2 Submissions*, 2020. [Online]. Available: https://csrc.nist.gov/Projects/Lightweight-Cryptography

[6] N. Mouha and B. Preneel, “Towards finding optimal parameters for SPONGENT,” in *Workshop on Lightweight Cryptography for Security and Privacy*, vol. 7, pp. 45–53, 2011.

[7] A. Poschmann, *Lightweight Cryptography: Cryptographic Engineering for a Pervasive World*, Ph.D. dissertation, Ruhr Univ. Bochum, 2009.

[8] National Institute of Standards and Technology (NIST), “Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process,” 2020. [Online]. Available: https://csrc.nist.gov/Projects/Lightweight-Cryptography

[9] S. Banik, T. Isobe, and Y. Sasaki, “Aegis and the NIST LWC Project: Design considerations,” *IACR Trans. Symmetric Cryptol.*, vol. 2020, no. 3, pp. 150–189, 2020. doi: 10.13154/tosc.v2020.i3.150-189.