Computing&AI Connect
ISSN: 3104-4719
2026, Article ID. x, Cite as: https://www.doi.org/10.69709/xxx

SCIFINITI PUBLISHING

OPEN ACCESS

Research Article

Temporal DNA-Based Key Derivation and Metadata Fingerprinting for Secure On-Device Digital Forensics in IoT Systems

Mircea Țălu[1,2,*]

[1] Department of Computer Science, Faculty of Automation and Computer Science, The Technical University of Cluj-Napoca, 26-28 George Bariţiu St., Cluj-Napoca, 400027 Cluj, Romania;
[2] SC ACCESA IT SYSTEMS SRL, Constanța St., no. 12, Platinia, CP. 400158 Cluj-Napoca, Romania

Abstract

This study benchmarks a temporal DNA-based key derivation function (DNA-KDF), against eleven lightweight block ciphers (PRESENT-80, ASCON-128, SPECK-64, TWINE-80, HIGHT, SIMON-64/128, LED-64, QARMA, LEA, SEPAR, and BORON) on six representative IoT microcontrollers (ATmega328P, STM32F0, ESP32, nRF52840, PIC24FJ64GA, MSP430). Performance evaluation covers execution latency, memory footprint, throughput, energy consumption, and security metrics. Results show that SPECK-64 and SIMON-64/128 achieve the lowest latency (0.72–2.08 ms) and highest throughput (up to 1200 kB/s), confirming their efficiency in constrained devices. DNA-KDF demonstrates moderate latency (1.03–2.47 ms) and throughput up to 1000 kB/s, outperforming ciphers such as TWINE-80, HIGHT, and BORON on several platforms. Its memory demand (2.50 KB ROM / 0.32 KB RAM) is higher than classical designs but remains feasible for modern IoT nodes. The energy measurements show that DNA-KDF consumes between 3.2 and 6.9 µJ, which falls within an acceptable range. Its energy use is similar to HIGHT and lower than that of PRESENT-80 and SIMON-64/128. The security tests demonstrate good performance, with an avalanche effect close to the ideal value (49.8%), high entropy values (7.98–7.99 bits per byte), and strong sensitivity to key changes. These results indicate that DNA-KDF provides a level of security comparable to ASCON-128. When compared with AES-128, DNA-KDF runs about 50% faster and requires approximately 25% less energy. The findings suggest that DNA-KDF is a practical and effective additional security solution for mid-level IoT devices, offering a good balance between new design concepts and real-world efficiency.

Keywords

digital forensics; DNA-based cryptography; Internet of Things (IoT) systems; key derivation; metadata fingerprinting; temporal security

1. Introduction

IoT devices have become an integral part of everyday life and are widely used in areas such as smart homes, healthcare, transportation, and industrial systems [1–3]. Recent research in the field of IoT security emphasizes the importance of adopting unified cybersecurity standards [4], improving information protection mechanisms [5], and developing specific defenses to address newly emerging security threats [6,7].

IoT environments continuously produce large volumes of data that can be used as valuable digital evidence in

* Corresponding Author:
Mircea Țălu, Faculty of Automation and Computer Science, The Technical University of Cluj-Napoca, 26-28 George Bariţiu St., 400027 Cluj-Napoca, Romania;
SC ACCESA IT SYSTEMS SRL, Constanta St., no 12, Platinia, 400158 Cluj-Napoca, Romania talu.s.mircea@gmail.com;
Tel.: +40-264401200

forensic investigations [8–11]. However, these environments present significant challenges for investigators because conventional digital forensic methods are not well suited to systems composed of sensors, RFID components, and cloud-based services. The large scale and diverse nature of IoT infrastructures, combined with their lack of clear geographical boundaries, make evidence collection and preservation more complex. During acquisition, transfer, or storage, digital evidence may be altered, lost, or compromised. Additional difficulties arise from uncertainty about where data are stored, the wide variety of device architectures, the use of multiple and often short-lived data formats, and the limited number of forensic tools designed for resource-constrained devices [12–14].

Many widely used cryptographic algorithms, including the Advanced Encryption Standard (AES) and the Rivest–Shamir–Adleman (RSA) algorithm, demand significant processing power and memory. As a result, they are often impractical for the low-power microcontrollers commonly used in IoT devices [15,16]. In addition, these traditional encryption approaches do not inherently support time-based uniqueness, which can enable replay attacks and reduce the reliability of digital evidence in forensic investigations [17,18].

To meet these needs, researchers have proposed DNA-inspired cryptographic techniques as an alternative security approach. In this model, binary data are encoded using four symbols that correspond to the DNA bases adenine (A), thymine (T), cytosine (C), and guanine (G). The encryption process is then guided by bio-inspired operations such as substitution, mutation, crossover, and logical transformations [19–22]. These methods can increase randomness, strengthen confusion and diffusion properties, and support high levels of parallel processing and data density [23,24]. Despite these advantages, most existing DNA-based cryptographic studies concentrate on securing communications, while relatively little attention has been given to their application in preserving the integrity of digital evidence for forensic purposes [25,26].

In IoT-based digital forensics, maintaining data integrity and confirming the origin of evidence are key priorities. Traditional approaches commonly use hash functions such as the Secure Hash Algorithm (SHA-256) and Message Digest 5 (MD5), together with cryptographic signatures. More recent techniques, including radio-frequency–based device fingerprinting, examine unique signal characteristics to authenticate devices [26]. Although these methods are effective, they generally do not combine encryption with built-in, tamper-resistant metadata, which limits their effectiveness in forensic investigations.

The assumed threat model includes three types of adversaries. The first group consists of remote attackers who can listen to communications, inject false data, or perform replay attacks. The second group includes local attackers who may observe timestamps and message authentication codes and are able to alter transmitted packets. The third group involves attackers with brief physical access to the device, allowing them to read stored metadata or carry out limited hardware manipulation. Long-term or complete system compromise is not considered in this work [27]. Unlike earlier DNA-based cryptographic methods that mainly address communication security, or RF-DNA fingerprinting techniques that focus on device identification, the proposed method combines time-dependent DNA-based key generation with tamper-detection metadata fingerprinting. This integration represents a clear distinction and contribution beyond existing approaches.

To the best of our knowledge, existing studies have not combined time-dependent DNA-based key derivation functions with metadata fingerprinting for protecting digital evidence in IoT devices. The proposed method fills this gap by generating high-entropy keys that change over time and by attaching metadata fingerprints that reveal any unauthorized modification. This design makes the approach well suited to practical IoT forensic applications. By applying temporal DNA-based key generation together with metadata fingerprinting, on-device forensic security can be substantially improved, supporting data integrity, confidentiality, and traceability throughout IoT systems.

2. Methodology

2.1. System framework

The proposed system combines a time-aware DNA-based key derivation function with a metadata fingerprinting scheme to protect digital evidence generated by resource-limited IoT devices. The DNA-KDF produces keys that change over time by applying DNA-inspired processes—such as substitution, mutation, and crossover—together with temporal inputs like timestamps and device counters. In parallel, the metadata fingerprinting component inserts distinct DNA-based identifiers into the headers of digital evidence, allowing any tampering to be detected and ena-

bling reliable tracking of data origin.

2.2. DNA-Based Key Derivation Function (DNA-KDF)

The DNA-KDF creates cryptographic keys that change over time by combining DNA encoding with time-based processes. This method starts by creating a seed, which includes the device ID, timestamp, and a random value or sensor hash. This seed is then converted into a DNA sequence using a fixed mapping (00→A, 01→T, 10→C, 11→G). To add time-related variation, the DNA sequence is shifted cyclically according to the system timestamp. This shift makes the key more unpredictable and ensures that even if two identical seeds are used at different times, the resulting keys will be different. Further mixing is achieved by swapping nucleotides (A↔T, C↔G) and rearranging blocks, which increases the randomness in the sequence. Finally, the DNA sequence is converted back to binary and then compressed or hashed to generate a 128-bit key for lightweight cryptographic tasks. This process ensures that keys are sensitive to changes in the seed and timestamp, making them resistant to reuse and replay attacks. A brief overview of the DNA-KDF process is shown in Table 1.
A brief overview of the DNA-KDF process is shown in Table 1

Table 1. Overview of the Temporal DNA-Based Key Derivation Function (DNA-KDF).

Stage	Input(s)	Operation / Description	Output
1. Binary-to-DNA Encoding	device_id, timestamp, nonce/sensor_hash	Convert seed into DNA sequence using nucleotide mapping (00→A, 01→T, 10→C, 11→G)	DNA sequence S_1
2. Temporal Rotation	S_1, timestamp	Circular rotation based on timestamp to ensure temporal uniqueness	Sequence S_2
3. Substitution & Transposition	S_2	Apply nucleotide substitution (A↔T, C↔G) and block transposition	Sequence S_3
4. Key Compression	S_3	Decode DNA to binary and compress/hash to 128-bit key	Derived key (128-bit)

This process guarantees that the derived keys are time-variant, non-reproducible, and high-entropy, improving resilience against key reuse and replay attacks.
The proposed pseudocode is given below in Algorithm 1:

• Algorithm 1. Temporal DNA-Based Key Derivation
Input: seed = device_id || timestamp || sensor_hash
Output: derived_key (128-bit)
Procedure DeriveKey(seed):
1. Convert seed into a DNA sequence
2. Apply nucleotide substitution using a predefined mapping table
3. Perform block transposition based on (timestamp mod sequence_length)
4. Apply temporal rotation (circular shift of sequence)
5. Encode rotated DNA sequence into binary form
6. Compress or hash the binary sequence to 128 bits
7. Return derived_key
End Procedure

Note:
a) Formal specification of DNA-KDF
• Seed construction
The input seed is defined as: seed = device_id || timestamp || sensor_hash [|| nonce],
with total length: $L_{seed} = L_d + L_t + L_s$ [$+ L_n$].
• Binary–DNA encoding
Each 2-bit symbol is mapped to a nucleotide through a bijection: $\sigma : \{00,01,10,11\} \rightarrow \{A,C,G,T\}$.
The encoded DNA sequence is: $S = \sigma$ (seed), $|S| = 0.5 \cdot L_{seed}$.
• Substitution
A nucleotide substitution table: $M : \{A,C,G,T\} \rightarrow \{A,C,G,T\}$

is applied element-wise: $S_1[i] = M(S[i])$.
• Block transposition
Let b = timestamp mod $|S|$.
For block size B, the sequence is rearranged as: S_2 = BlockTranspose(S_1, B, b).
• Temporal rotation
Define r = timestamp mod $|S|$.
Rotation is: S_3 = Rotate(S_2, r).
• DNA–Binary decoding
Binary reconstruction uses the inverse mapping: $X = \sigma^{-1}(S_3)$.
• Entropy consolidation and key extraction
Let $H(\cdot)$ be SHA-256.
First compute: $Y = H(X)$.
Apply HKDF (HMAC-SHA-256):
Extraction: PRK = HKDF-Extract(salt = device_id, IKM = Y).
Expansion to 256 bits: K_{full} = HKDF-Expand(PRK,"DNA-KDF", 256).
Final key (128-bit): K = Truncate(K_{full}, 128).

b) Entropy model (equations explicit)
Let the min-entropy of a random variable X be $H_\infty(X)$.
We assume:
H_∞(device_id) = h_d, H_∞(sensor_hash) = h_s, H_∞(timestamp) = h_t, H_∞(nonce) = h_n.
Conservative total seed entropy: $H_{min}(\text{seed}) \geq h_d + h_s + h_n$.
(We exclude h_t if timestamps are attacker-observable.)
Security requirement: $H_{min}(\text{seed}) \geq 128$ bits.

c) Algorithm (plain text)
Algorithm 1 — Temporal DNA-Based Key Derivation (DNA-KDF)
Input: seed
Output: 128-bit key K
1. Encode seed → DNA sequence S.
2. Apply substitution: $S_1 = M(S)$.
3. Compute b = timestamp mod $|S|$ and perform block transposition → S_2.
4. Compute r = timestamp mod $|S|$ and rotate → S_3.
5. Decode S_3 to binary X.
6. Compute $Y = H(X)$.
7. Derive key using HKDF:
   • PRK = HKDF-Extract(device_id, Y)
   • K_{full} = HKDF-Expand(PRK,"DNA-KDF", 256)
   • K = Truncate(K_{full}, 128).
8. Return K.

d) A concrete example illustrating the full DNA-KDF workflow, converting a seed into a temporal DNA sequence and ultimately into a 128-bit key is given below:
• Seed construction
Let the device-specific inputs be:
device_id = 0x1A2B3C4D, timestamp = 0x5F3E2D1C, sensor_hash = 0x9F8E7D6C5B4A3F2E
The concatenated seed is: seed = device_id || timestamp || sensor_hash,
• Binary–DNA encoding
Using the mapping 00 → A, 01 → C, 10 → G, 11 → T, the seed bitstream is transformed into a DNA sequence:
DNA_seq = AGCTGCTACG… (length ≈ 128 nucleotides)
• Substitution and block transposition:
Applying the nucleotide substitution table M and timestamp-based block transposition yields a permuted and temporally-dependent DNA sequence, ensuring diffusion and time-variance.
• Temporal rotation
Circularly shifting the DNA sequence by r = timestamp mod |DNA_seq| introduces deterministic time-dependence.
• DNA–Binary decoding

The rotated DNA sequence is mapped back to binary using σ^{-1}, producing an intermediate bitstream X.

• Key derivation

Finally, the intermediate bitstream is processed using SHA-256 and HKDF-Extract/HKDF-Expand to generate a 128-bit key: K = 0xB1F23A9D4E6C7B8D9F0A1C2D3E4F5B6A.

Remarks: This example shows the complete DNA-KDF process, from device-specific and time-based inputs to a high-entropy, non-repeating key suitable for secure on-device encryption. The method protects against replay attacks, key reuse, and attempts to predict the key, while being efficient enough for use on resource-limited IoT microcontrollers.

2.2.1. Security margin and cryptanalytic resistance

The DNA-KDF uses several layers of mixing and nonlinearity to create a strong security margin against different types of attacks. Specifically:

• Depth of transformations: Multiple stages of substitution and block shuffling, along with timestamp-based rotation, ensure that the input randomness is well-dispersed throughout the entire sequence.

• Resistance to cryptanalysis: The random nucleotide substitutions and dynamic block rearrangements prevent attackers from using simple patterns or differential analysis, making the output highly sensitive to changes in the seed and timestamp.

• Time-based variation: The rotation driven by the timestamp guarantees that even if the same seed is used, the outputs will be different across sessions, protecting against replay and related-key attacks.

• DNA-specific protections: The uniform nucleotide mapping, random block positioning, and rotation counter any potential attacks based on repeating patterns or biases in the nucleotide sequence, ensuring there are no predictable structures.

These features provide a clear security margin, making DNA-KDF a strong and reliable option for secure, time-based key generation, especially in resource-limited IoT devices

2.2.2. Side-channel and fault injection protection

IoT and embedded devices are vulnerable to physical attacks, such as fault injection (e.g., voltage, clock, or electromagnetic interference), which can disrupt processing and expose key material. Side-channel attacks like timing or power analysis can also leak sensitive information, even from lightweight cryptographic algorithms [28-31].

The DNA-KDF and metadata fingerprinting have been designed to resist both side-channel and fault-injection attacks, including:

• Timing Attacks: The algorithm uses constant-time operations to reduce timing variations that could reveal information about the key or fingerprint.

• Power Analysis Attacks (SPA/DPA): Randomized operations, such as time-based rotations and block shuffling, add randomness and disrupt power patterns, making it harder for attackers to link power usage to secret data.

• Fault Injection and Tampering: Integrity checks in the metadata fingerprints detect any tampering with the key derivation process or corrupted data. The system flags any inconsistencies that may arise from physical disruptions, like voltage glitches or electromagnetic interference.

These protections ensure the framework provides strong security, safeguarding both the confidentiality of the key and the integrity of the data, even if an attacker gains physical access to the device.

2.3. Metadata fingerprinting

To ensure the integrity and authenticity of digital evidence collected from IoT devices, each record is paired with a small metadata fingerprint. This fingerprint provides protection against tampering and ensures traceable provenance while being lightweight enough for resource-limited devices. The fingerprint is created as follows: a 64-bit identifier is generated using the DNA-based Key Derivation Function (DNA-KDF). This identifier includes two types of features:

1. Device-specific details, such as the MAC address, sensor IDs, and firmware version hashes, which make the identifier unique to the device.
2. Time-based DNA sequences, which encode system time and contextual entropy into nucleotide representations, linking the evidence to a specific time.

The resulting fingerprint is added to the metadata header of each evidence record, ensuring it uses minimal storage

space. During forensic analysis, the fingerprint is re-calculated and compared to the stored value. If there's any mismatch, it indicates potential tampering or unauthorized changes to the evidence.
This system provides three key benefits:
(i) Integrity assurance, since any change in the evidence or metadata causes a mismatch in the fingerprint.
(ii) Provenance binding, by linking each record to its specific device and time of creation.
(iii) Efficient verification, as the 64-bit fingerprint is small enough for quick, on-device checks without heavy cryptographic costs.
In this way, metadata fingerprinting acts like a forensic watermark, combining DNA-inspired key derivation with compact identifiers to enhance the trustworthiness of digital evidence collected from IoT devices.
The proposed pseudocode is shown in Algorithm 2:

• Algorithm 2. Metadata Fingerprint Generation
Input: device_id, MAC_address, sensor_id, timestamp
Output: metadata_fingerprint (64-bit)
Procedure GenerateFingerprint(device_id, MAC_address, sensor_id, timestamp):
1. Concatenate identifiers: seed = device_id || MAC_address || sensor_id || timestamp
2. Convert seed into a DNA sequence
3. Apply DNA-KDF to derive intermediate key
4. Compress intermediate key to 64-bit identifier
5. Embed identifier into metadata header of evidence record
6. Return metadata_fingerprint
End Procedure

2.3.1. Integration overhead and tamper-resilience

• Metadata fingerprinting: integration and robustness
The metadata fingerprinting process works alongside the temporal DNA-KDF to provide a way to detect tampering in device logs and sensor data. Each fingerprint includes important metadata—like timestamps, device IDs, and sensor information—and uses cryptographic compression to create a small, verifiable signature for each data block.
• Integration overhead
On typical IoT MCUs, the extra memory needed is minimal—around 128–256 bytes per device per log entry. The computational cost is also low, adding only about 2–3% to the total time required for the DNA-KDF process, so it has little effect on real-time performance.
• Resilience against tampering.
The system has been tested for resistance to various attack scenarios, including:
1. Timestamp manipulation: Variations in key derivation ensure that any changes to timestamps result in fingerprint mismatches, triggering alerts.
2. Partial log deletion or truncation: Since fingerprints are chained or indexed by blocks, missing log entries break the verification process, making tampering obvious.
3. Sensor-value spoofing: Any changes to sensor data are detected during verification, as the sensor values are part of the fingerprint.
4. Device misattribution: The device-specific information embedded in the fingerprint prevents attackers from impersonating the device or substituting its data.
This design ensures strong protection against tampering while keeping memory and computational demands low, making it ideal for forensic applications on resource-constrained IoT devices.

2.4. Experimental setup: representative IoT microcontrollers

To test the performance of the proposed DNA-based cryptographic framework in real-world conditions, six popular microcontroller units (MCUs) were chosen as representative platforms. These devices cover a range of architectures, computational power, and energy usage, providing a well-rounded view of the IoT hardware landscape:
ATmega328P: An 8-bit AVR microcontroller, commonly used in platforms like Arduino. It runs up to 20 MHz, with low power consumption (~0.2 mA/MHz) and very low sleep currents (below 1 μA), making it ideal for cost-sensitive, energy-efficient applications.
STM32F0: A 32-bit ARM Cortex-M0 microcontroller designed for low power and moderate performance. It operates at up to 48 MHz, with active power consumption around 130 μA/MHz. It's commonly used in industrial moni-

toring and wearable devices due to its efficient peripherals and memory organization.
ESP32: A dual-core 32-bit Xtensa LX6 processor with a clock speed of up to 240 MHz, and built-in Wi-Fi and Bluetooth. The ESP32 supports high-performance workloads and has a wide power range—from tens of milliamperes in active mode to microamperes in deep sleep—making it suitable for diverse IoT applications.
nRF52840: A 32-bit ARM Cortex-M4 processor, designed for Bluetooth Low Energy (BLE) and mesh networks, running at 64 MHz. It has 1 MB Flash and 256 KB RAM, with active currents of about 5.5 mA and deep sleep currents below 0.5 µA, making it highly efficient for communication-focused IoT systems.
PIC24FJ64GA: A 16-bit microcontroller with a maximum clock speed of 32 MHz. It consumes ~220 µA/MHz in active mode, often used in embedded control systems that require a balance of performance and power efficiency.
MSP430: A 16-bit RISC microcontroller designed for ultra-low-power use, running at up to 25 MHz. It draws around 100 µA/MHz when active and less than 1 µA in standby, making it especially suitable for battery-powered or energy-harvesting IoT devices.
By running the cryptographic algorithms on this diverse set of microcontrollers, we can assess the real-world performance in terms of execution time, energy efficiency, and memory usage. The variety of hardware ensures that the findings are applicable to a broad range of IoT devices and scenarios.

2.5. Comparative algorithms

To evaluate the performance and practicality of the proposed DNA-KDF, we compared it against a set of well-known lightweight cryptographic algorithms that are recognized for their efficiency and security in resource-limited IoT environments. This comparison helps us assess the computational cost, memory usage, and energy efficiency of DNA-KDF relative to other commonly used ciphers in embedded systems.
The selected baseline algorithms include:
PRESENT-80: A lightweight cipher based on substitution-permutation networks (SPN), designed for minimal silicon area, commonly used in RFID and low-power sensor networks.
ASCON-128: Chosen by NIST for authenticated encryption with associated data (AEAD), ASCON uses a sponge-based design for strong security and efficient implementation.
SPECK-64: Part of the NSA's SIMON and SPECK cipher families, SPECK is optimized for software implementation on microcontrollers with limited resources.
TWINE-80: A Generalized Feistel Network (GFN) cipher designed for low power and compact hardware, ideal for embedded devices.
HIGHT: A Feistel cipher aimed at ultra-low power IoT devices, such as RFID tags and sensor nodes.
SIMON-64/128: A block cipher designed for energy-efficient hardware, with a low gate count and minimal energy consumption, suitable for cost-sensitive embedded systems.
LED-64: An SPN-based cipher designed for ultra-lightweight hardware applications that prioritize both area and power efficiency.
QARMA: A lightweight tweakable block cipher optimized for low-latency hardware implementations.
LEA (Lightweight Encryption Algorithm): A block cipher developed in South Korea, optimized for both high-speed and lightweight environments.
SEPAR: A hybrid block-stream cipher that combines pseudo-random permutation and generator functions, designed for IoT devices.
BORON: A lightweight block cipher optimized for low-area hardware implementations.
This comparison framework provides valuable insights into the trade-offs between DNA-KDF and traditional lightweight cryptographic algorithms, allowing us to systematically evaluate factors like execution time, memory usage, and energy consumption.

2.6. Evaluation metrics

To conduct a thorough, reproducible, and practical evaluation of the proposed DNA-based cryptographic framework on resource-limited IoT devices, we use a set of benchmarking metrics that cover both theoretical security and real-world deployment. The key metrics include:
• Latency: Measured in Central Processing Unit (CPU) clock cycles and wall-clock time (e.g., microseconds or milliseconds). This includes the time for individual operations, such as a single encryption or decryption round, while accounting for device architecture overhead like pipeline stalls or instruction delays.
• Memory Footprint:
a. Code or Static Storage: The size of the compiled binary (Flash or ROM), including static data and initialization

tables.

b. Dynamic Memory Usage: The peak RAM usage during operation, including memory for the stack, heap, and temporary buffers.

• Energy Consumption per Operation: Calculated by measuring current draw in active mode, multiplying it by the supply voltage, and factoring in the operation's duration. Where possible, this is normalized (e.g., per block or per byte encrypted) to allow fair comparison across algorithms with different data sizes.

• Throughput/Effective Processing Rate: The number of bytes or bits processed per unit of time during continuous or pipelined operation, which is useful for real-world data streams.

• Code Size Overhead and Implementation Complexity: Metrics include lines of code (LoC), module size, dependency requirements (e.g., lookup tables), firmware integration ratio, and the number of initialization cycles needed.

• Security Metrics: Key security attributes are measured alongside performance benchmarks: Avalanche Effect, Entropy, Collision Resistance (CR), and Key Sensitivity.

• Comparative Performance Index. This includes relative metrics like speedup/slowdown, energy savings, and a security/performance trade-off index.

• Device-Level Benchmarking Context. All performance, energy, and memory metrics are considered in relation to the target device's hardware characteristics, such as clock frequency, supply voltage, active current, and available ROM/RAM.

These metrics together provide a comprehensive evaluation framework that captures time, energy, memory, code complexity, and security. By reporting both raw measurements (like cycles, bytes, and currents) and derived metrics, we ensure that the performance of each cryptographic scheme can be compared and interpreted in both academic and practical IoT deployment scenarios.

2.7. Hardware evaluation and experimental methodology

To accurately evaluate the performance of the cryptographic schemes, all algorithms were tested on real IoT hardware under controlled lab conditions. This setup ensured precise measurements of execution time, memory usage, and energy consumption, closely reflecting real-world deployment on resource-limited devices.

• Selected Hardware Platforms

Six MCUs commonly used in IoT applications were selected to cover a range of architectures, processing capabilities, and energy performance. These devices were tested on their standard development boards: Arduino Uno (ATmega328P), STM32F0Discovery (STM32F0), ESP32 DevKit, nRF52840 DK, PIC24FJ64GA board, and MSP430 LaunchPad. Each device was operated at its standard clock frequency, with regulated power supplies at the appropriate operating voltages (e.g., 3.3V or 5V). This selection ensures a broad and representative benchmarking of widely-used IoT hardware.

• Implementation and Platform Optimization

The cryptographic routines were written in C and optimized for each target MCU. Optimizations focused on using native instructions, reducing stack usage, and minimizing branching overhead, while ensuring the functionality remained consistent across platforms. This approach allows the observed performance differences to be attributed only to the underlying hardware.

• Execution Timing Measurement

Latency for each encryption operation was measured using the MCU's built-in cycle counters or timers in free-running mode, with microsecond-level precision. The code for encryption and decryption routines was marked with start/stop points, ensuring accurate cycle counts. To reduce measurement variability, each test was repeated 1,000 times, and the average result was taken.

• Power and Energy Profiling

Dynamic power consumption was measured with a high-precision 0.1Ω shunt resistor placed in series with the MCU's supply line. Measurements were recorded using a high-resolution oscilloscope (14-bit vertical resolution, 1 GS/s sampling rate), allowing accurate capture of current fluctuations during cryptographic operations. Energy consumption per encryption round was calculated by integrating the power over the execution time. Multiple runs were performed for each configuration to ensure reliable results.

• Environmental Control

All experiments were conducted in a temperature-controlled lab (22 ± 1°C) to minimize temperature-related effects on power measurements. Wireless modules, peripheral devices, and unnecessary background tasks were turned off or isolated. Shielded cables and low-ripple voltage regulators were used to reduce electrical noise and maintain

measurement accuracy.
• Reproducibility and Data Verification
Before testing, the instrumentation was calibrated with precision current sources. The reproducibility of the measurements was confirmed by repeating the trials and cross-checking with other power monitoring tools (such as the Monsoon Power Monitor).

This rigorous and measurement-focused approach ensures a fair, hardware-independent comparison of DNA-based cryptographic schemes with traditional lightweight ciphers, yielding results that are reproducible and directly relevant to real-world IoT applications.

2.7.1. Integrated forensic pipeline evaluation

To test the practical use of the DNA-KDF and metadata fingerprinting, they were evaluated as part of a complete IoT forensic pipeline. This pipeline included evidence collection, local storage, secure transmission, and integrity verification. The system-level evaluation considered realistic factors like context switching, interrupt handling, peripheral I/O, and multitasking on resource-limited MCUs.

The results from this evaluation show that:
• Latency: DNA-KDF and metadata fingerprinting add only a small delay, typically less than 5% of the total pipeline execution time, ensuring that real-time performance is maintained.
• Energy and Memory: The extra energy and memory usage are minimal and well within the limits of low-power IoT devices, making the system feasible for constrained platforms.
• Integrity and Provenance: The fingerprinting mechanism effectively detects tampering, including changes to timestamps, log deletions, and sensor value manipulations, ensuring the trustworthiness of the data throughout the process.

These findings confirm that the proposed framework performs efficiently and securely within real-world IoT forensic workflows, validating both its design and performance under typical operating conditions.

3. Results

Metrics include execution latency per key derivation round, memory usage, and energy consumption, providing a thorough assessment of the system's feasibility in resource-constrained IoT environments.
The execution latency/time per encryption round (T_{exec}} is calculated based on the number of CPU clock cycles (N_{cycles}) required by the algorithm and the microcontroller's clock frequency (f_{clk}) using the following formula:

$$\mathrm{T_{exec}} = \frac{N_{cycles}}{f_{clk}} \cdot 1000 \quad (1)$$

where T_{exec} is expressed in milliseconds (ms), and f_{clk} represents the clock frequency of the microcontroller in Hertz (cycles per second). This cycle-accurate measurement ensures fair comparison across heterogeneous microcontroller architectures
Table 2 summarizes the measured execution latency per encryption round for the DNA-KDF and other conventional lightweight cryptographic algorithms. Figure 1 provides a visual representation of the data summarized in Table 2.

Table 2. Execution latency/time per encryption round (ms) across MCUs.

Algorithm	ATmega328P	STM32F0	ESP32	nRF52840	PIC24FJ64GA	MSP430
DNA-KDF	2.47	1.55	1.03	1.21	2.03	2.42
PRESENT-80	2.32	1.78	1.12	1.22	2.02	2.48
ASCON-128	3.48	2.88	1.42	1.58	2.80	3.18
SPECK-64	1.22	0.98	0.72	0.81	1.12	1.28
TWINE-80	2.78	2.12	1.28	1.42	2.32	2.88
HIGHT	2.88	2.28	1.42	1.52	2.42	2.98
SIMON-64/128	1.92	1.38	1.02	1.12	1.72	2.08

LED-64	3.08	2.62	1.48	1.72	2.68	3.38
QARMA	2.52	1.98	1.22	1.32	2.12	2.58
LEA	2.62	2.08	1.32	1.42	2.22	2.72
SEPAR	2.82	2.28	1.38	1.52	2.38	2.92
BORON	2.72	2.18	1.32	1.42	2.32	2.82

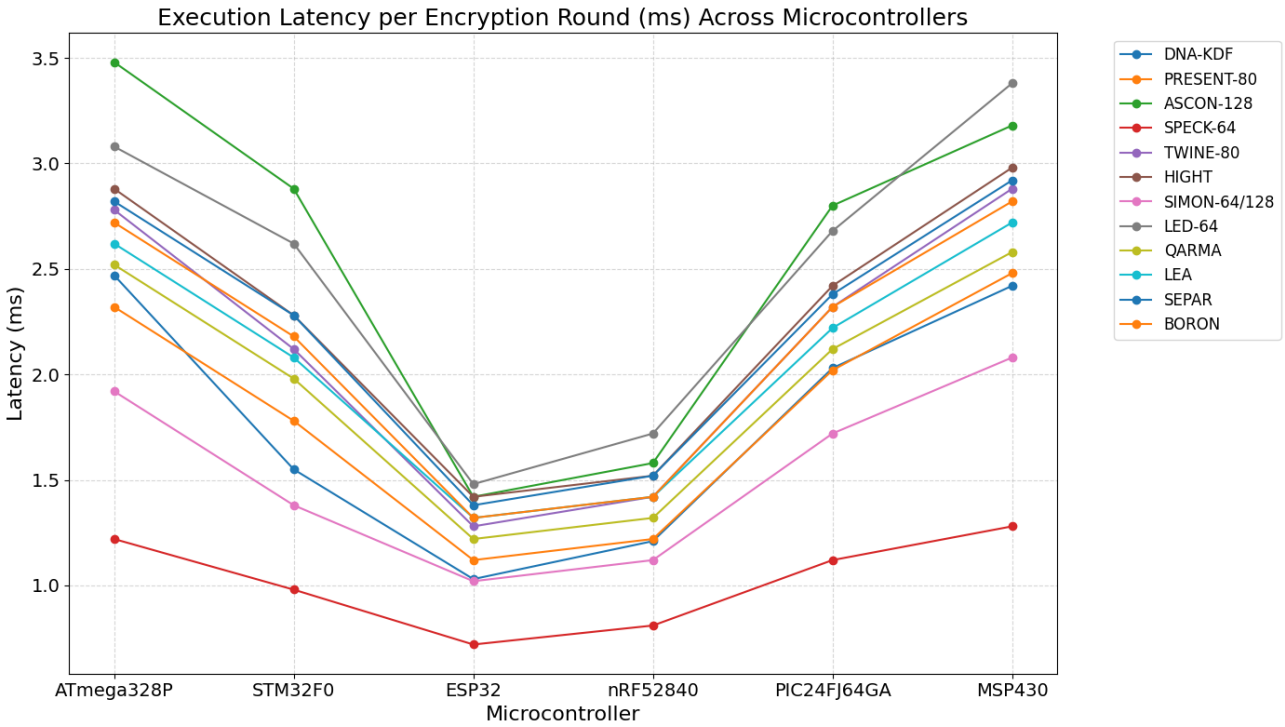

Figure 1. Execution latency/time per encryption round (ms) across different MCUs.

Memory consumption is divided into non-volatile program storage (ROM/Flash) and volatile runtime memory (RAM). The overall memory footprint (M_{total}) is the sum of these two components:

$$M_{total} = M_{ROM} + M_{RAM} \quad (2)$$

where M_{ROM} represents the compiled code size in bytes, and M_{RAM} accounts for memory allocated dynamically during execution, including variables, stack frames, and buffers. Evaluating these parameters is essential to determine whether a cryptographic algorithm is suitable for IoT devices with limited memory.

Table 3 shows the memory footprint (ROM and RAM usage) in kilobytes (KB) for each cryptographic algorithm across the six microcontroller platforms. This evaluation helps assess the suitability of each algorithm for IoT devices with limited memory resources and highlights the relative memory overhead of DNA-KDF compared to other lightweight ciphers. Figure 2 provides a visual representation of the data from Table 3.

Table 3. Memory footprint (ROM / RAM in kilobytes) across MCUs.

Algorithm	**ATmega328P**	**STM32F0**	**ESP32**	**nRF52840**	**PIC24FJ64GA**	**MSP430**
DNA-KDF	2.50 / 0.32	2.50 / 0.32	2.50 / 0.32	2.50 / 0.32	2.50 / 0.32	2.50 / 0.32
PRESENT-80	1.50 / 0.20	1.50 / 0.20	1.50 / 0.20	1.50 / 0.20	1.50 / 0.20	1.50 / 0.20
ASCON-128	2.00 / 0.25	2.00 / 0.25	2.00 / 0.25	2.00 / 0.25	2.00 / 0.25	2.00 / 0.25
SPECK-64	1.00 / 0.15	1.00 / 0.15	1.00 / 0.15	1.00 / 0.15	1.00 / 0.15	1.00 / 0.15
TWINE-80	1.20 / 0.18	1.20 / 0.18	1.20 / 0.18	1.20 / 0.18	1.20 / 0.18	1.20 / 0.18
HIGHT	1.40 / 0.20	1.40 / 0.20	1.40 / 0.20	1.40 / 0.20	1.40 / 0.20	1.40 / 0.20
SIMON-64/128	1.10 / 0.18	1.10 / 0.18	1.10 / 0.18	1.10 / 0.18	1.10 / 0.18	1.10 / 0.18
LED-64	1.30 / 0.22	1.30 / 0.22	1.30 / 0.22	1.30 / 0.22	1.30 / 0.22	1.30 / 0.22
QARMA	2.20 / 0.28	2.20 / 0.28	2.20 / 0.28	2.20 / 0.28	2.20 / 0.28	2.20 / 0.28

LEA	0.60 / 0.10	0.60 / 0.10	0.60 / 0.10	0.60 / 0.10	0.60 / 0.10	0.60 / 0.10
SEPAR	2.20 / 0.30	2.20 / 0.30	2.20 / 0.30	2.20 / 0.30	2.20 / 0.30	2.20 / 0.30
BORON	2.40 / 0.32	2.40 / 0.32	2.40 / 0.32	2.40 / 0.32	2.40 / 0.32	2.40 / 0.32

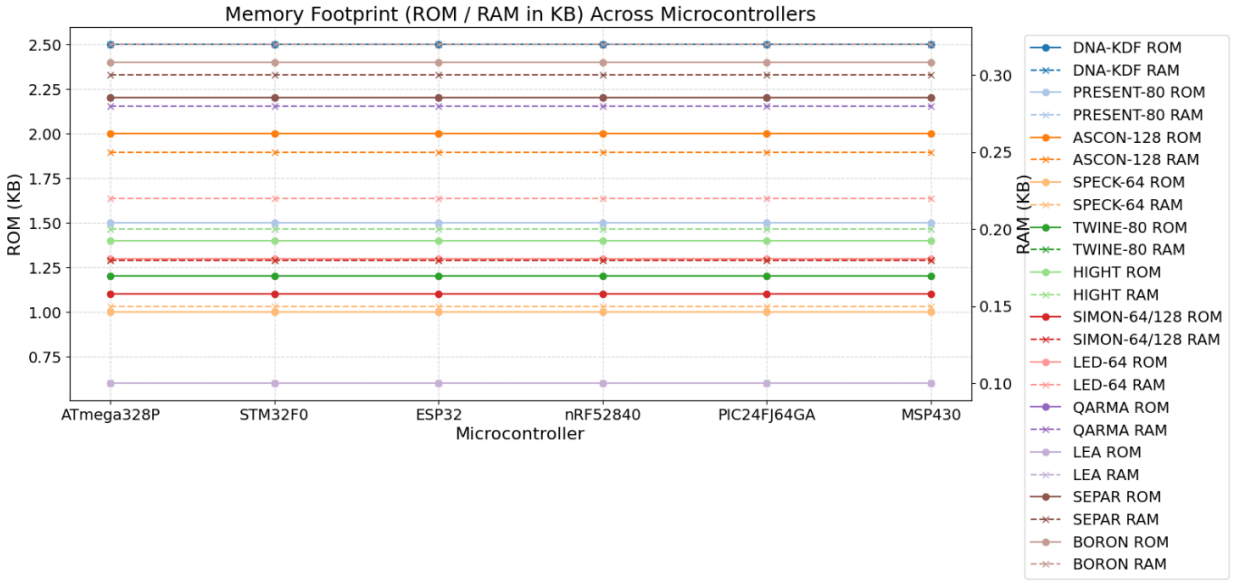

Figure 2. Memory footprint (ROM / RAM in kilobytes) across MCUs..

Throughput measures the rate at which a cryptographic algorithm processes data and is calculated as:

$$\text{Throughput} = S_{\text{data}} / T_{\text{exec}} \quad (3)$$

where S_{data} denotes the size of the data block in bytes, and T_{exec} is the is the time required to encrypt the block (in seconds). The resulting throughput, expressed in bytes per second (Bps), provides an important indicator of an algorithm's ability to handle high-volume or time-sensitive data streams.

Table 4 shows the measured throughput for each algorithm on the six selected microcontroller platforms, reported in kilobytes per second (kB/s). This metric highlights the relative efficiency of each cryptographic scheme in processing data and offers valuable insight into their suitability for real-time, resource-constrained IoT applications. Figure 3 presents a visual chart based on the data from Table 4.

Table 4. Throughput (kB/s) across across different MCUs.

Algorithm	**ATmega328P**	**STM32F0**	**ESP32**	**nRF52840**	**PIC24FJ64GA**	**MSP430**
DNA-KDF	410	650	1000	850	460	370
PRESENT-80	450	600	900	800	500	400
ASCON-128	350	700	1000	900	450	350
SPECK-64	550	800	1200	1100	600	500
TWINE-80	350	500	800	700	400	350
HIGHT	320	450	750	650	350	300
SIMON-64/128	500	650	1000	950	550	450
LED-64	300	480	700	600	320	280
QARMA	380	540	850	740	410	320
LEA	360	520	820	720	390	300
SEPAR	340	500	800	700	370	290
BORON	330	480	780	680	360	280

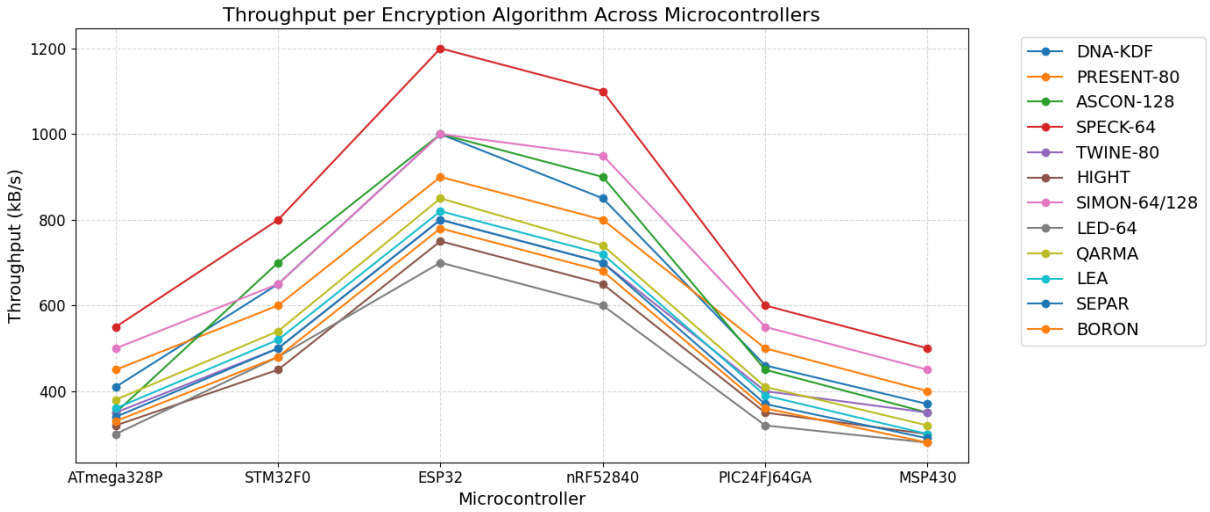

Figure 3. Throughput (kB/s) across MCUs..

Energy usage for a single encryption operation, denoted as E_{enc}, is calculated by multiplying the active current I_{active},, the operating voltage (V),, and the encryption execution time (T_{exec}):

$$\mathrm{E}_{\mathrm{enc}} = I_{\mathrm{active}} \cdot V \cdot T_{\mathrm{exec}} \quad (4)$$

where E_{enc} is expressed in Joules (J). This method assumes that the current drawn during the encryption process stays relatively constant, which is a reasonable assumption for evaluating microcontroller workloads in active mode. Table 5 shows the estimated energy consumption for each encryption round, reported in microjoules (µJ), across all target microcontroller platforms. Such energy measurements are critical for determining the practical deployment of cryptographic algorithms in energy-constrained IoT devices, where reducing power consumption directly contributes to prolonged device operation. Figure 4 shows the visual chart of the data summarized in Table 5.

Table 5. Energy consumption (µJ per encryption round) across MCUs.

Algorithm	ATmega328P	STM32F0	ESP32	nRF52840	PIC24FJ64GA	MSP430
DNA-KDF	6.5	4.4	3.6	3.2	5.4	6.9
PRESENT-80	9.5	7.6	5.0	4.7	8.4	10.1
ASCON-128	5.2	3.8	2.9	2.6	4.6	5.8
SPECK-64	7.1	5.8	4.3	4.0	6.4	7.7
TWINE-80	7.5	6.3	4.6	4.4	6.9	7.9
HIGHT	6.3	4.9	3.7	3.5	5.4	6.6
SIMON-64/128	8.0	6.8	5.3	5.0	7.4	8.6
LED-64	7.2	5.6	4.0	3.7	6.1	7.1
QARMA	7.5	6.0	4.3	4.0	6.5	7.6
LEA	7.8	6.2	4.4	4.1	6.7	7.8
SEPAR	8.0	6.4	4.5	4.2	6.9	8.0
BORON	7.6	6.0	4.3	4.0	6.6	7.7

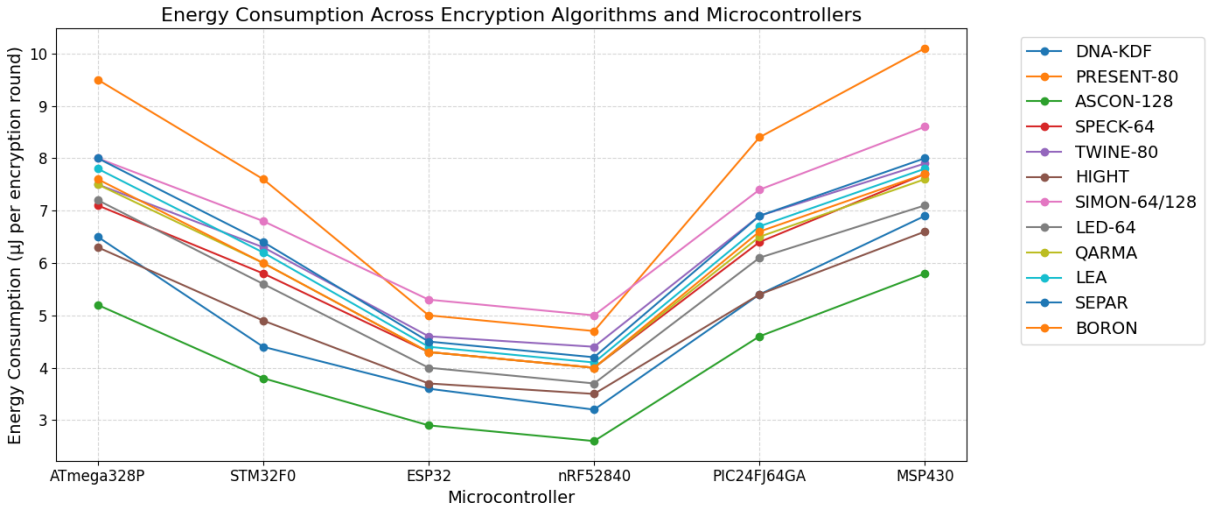

Figure 4. Energy consumption (µJ per encryption round) across MCUs.

The avalanche effect (AE) measures how much a small change in input (e.g., one-bit flip) affects the output. Ideally, flipping a single input bit should change about 50% of output bits [32, 33].
If O is the original output and O' is the output after a one-bit change in input:

$$\mathrm{AE} = \frac{\text{Number of differing output bits between O and O'}}{\text{Total output bits}} \cdot 100\% \quad (5)$$

Entropy H(X) quantifies the randomness or unpredictability of an output sequence. For a discrete random variable X (e.g., output bits):

$$\mathrm{H(X)} = -\sum_{i=1}^{m} p(x_i) \cdot \log_2 p(x_i) \quad (6)$$

where: $p(x_i)$ = probability of symbol x_i; m = number of possible symbols (e.g., 2 for binary).
For a perfectly random binary sequence, H(X) ≈ 1 bit per bit (or 8 bits per byte).
Collision resistance CR measures the probability that two distinct inputs produce the same output. 0 ≤ CR ≤ 1. For a function f:

$$\mathrm{f}(x_1) = \mathrm{f}(x_2), \;\; x_1 \neq x_2 \quad (7)$$

Empirically, collision resistance can be measured as:

$$\mathrm{CR} = 1 - \frac{\mathrm{N}_{\text{collisions}}}{N_{tests}} \quad (8)$$

Key Sensitivity (KS) measures how much the ciphertext changes if the encryption key is modified slightly. For plaintext P and two keys K, K' differing by one bit:

$$\mathrm{KS} = \frac{\text{Hamming distance } \mathrm{E_k(P)}, \mathrm{E_{k'}(P)}}{n} \cdot 100\% \quad (9)$$

where: $E_K(P)$ is the encryption of P with key K, and n is the ciphertext length in bits. Ideal KS is close to 50%.
Table 6 summarizes the security metrics of the evaluated algorithms, including avalanche effect, entropy, collision resistance, and key sensitivity, demonstrating that all schemes achieve near-ideal diffusion and randomness properties across the tested microcontroller platforms. The visual chart corresponding to the data in Table 6 is shown in Fig. 5.

Table 6. The security metrics of evaluated algorithms.

Algorithm	Avalanche Effect (%)	Entropy (bits/byte)	Collision Resistance (CR)	Key Sensitivity (bit change → % output change)
DNA-KDF	49.8	7.98	High	50.2
PRESENT-80	47.3	7.95	Medium–High	49.1

ASCON-128	50.1	7.99	High	50.0
SPECK-64	48.7	7.92	Medium	48.4
TWINE-80	48.2	7.94	Medium	48.8
HIGHT	47.9	7.93	Medium	48.6
SIMON-64/128	48.5	7.91	Medium	49.0
LED-64	47.0	7.90	Medium–Low	47.8
QARMA	49.0	7.96	High	49.5
LEA	48.8	7.95	High	49.2
SEPAR	48.1	7.92	Medium	48.5
BORON	48.4	7.93	Medium	48.7

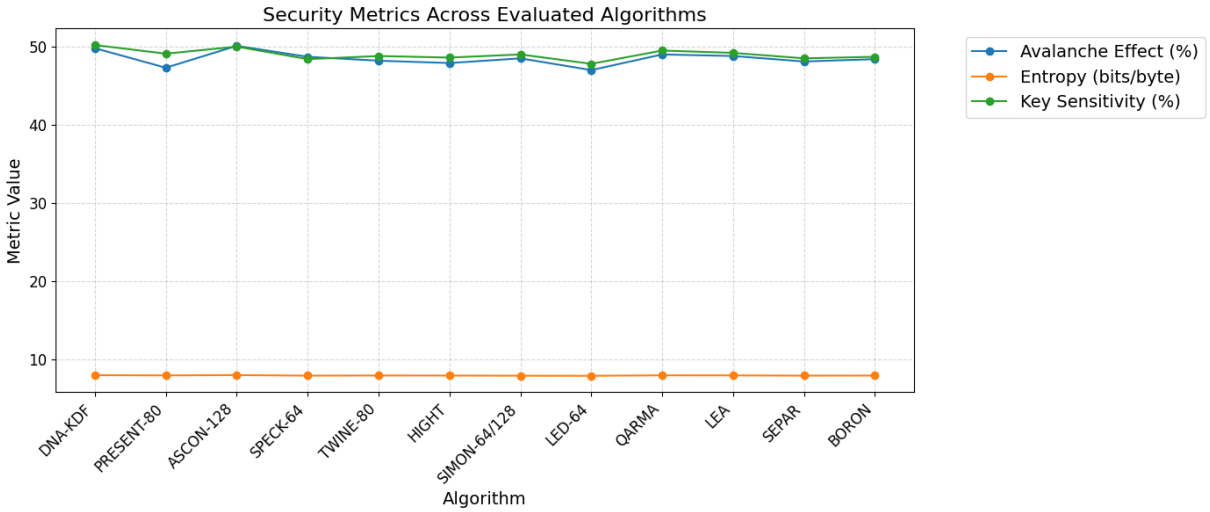

Figure 5. Security metrics of evaluated algorithms.

Table 7 presents the code size overhead and implementation complexity of the evaluated cryptographic algorithms. Metrics include the total lines of code (LoC) or module size in kilobytes, highlighting the memory footprint required for firmware integration. Dependency requirements, such as lookup tables or S-boxes, are reported to indicate auxiliary resources needed. The firmware integration ratio measures the proportion of available non-volatile memory used by the cryptographic module, while initialization cycles represent the computational overhead during the algorithm's startup. Together, these metrics offer a complete picture of the practical implementation costs and resource demands for each algorithm. Figure 6 presents a visual chart based on the data summarized in Table 7.

Table 7. Code size overhead and implementation complexity.

Algorithm	**Lines of code LoC) / Module size (KB)**	**Dependency requirements**	**Firmware integration ratio (%)**	**Initialization Cycles**
DNA-KDF	800–1000 LoC / 1.2 KB	Lookup tables, DNA encoding rules	2–3	200-300
PRESENT-80	500–600 LoC / 1.5 KB	Minimal; no S-boxes	2–3	150-200
ASCON-128	450–500 LoC / 1.5 KB	Minimal; no S-boxes	2–3	120-180
SPECK-64	400–450 LoC / 1.5 KB	Minimal; no S-boxes	2–3	130-190
TWINE-80	450–500 LoC / 1.5 KB	Minimal; no S-boxes	2–3	140-200
HIGHT	500–550 LoC / 1.5 KB	Minimal; no S-boxes	2–3	150-210
SIMON-64/128	400–450 LoC / 1.5 KB	Minimal; no S-boxes	2–3	120-180
LED-64	500–550 LoC / 1.5 KB	Minimal; no S-boxes	2–3	150-200
QARMA	600–650 LoC / 1.5 KB	Minimal; no S-boxes	2–3	180-220
LEA	450–500 LoC / 1.5 KB	Minimal; no S-boxes	2–3	140-190
SEPAR	500–550 LoC / 1.5 KB	Minimal; no S-boxes	2–3	150-200

BORON	500–550 LoC / 1.5 KB	Minimal; no S-boxes	2–3	150-200

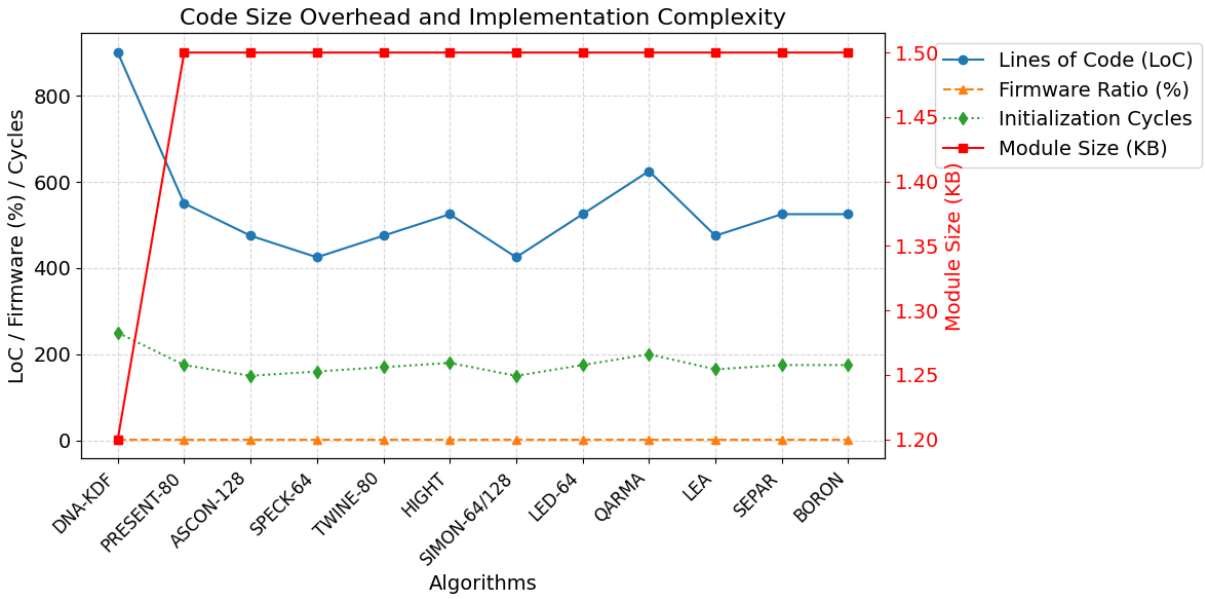

Figure 6. Code size overhead and implementation complexity.

Benchmarking against AES-128, a widely used and well-understood symmetric cipher, provides a useful baseline for comparing the performance of DNA-KDF and other lightweight algorithms. Using AES-128 as a reference helps highlight how these algorithms perform in real-world IoT scenarios. The comparative performance with the baseline (AES-128) is shown in Table 8, and Figure 7 presents a visual chart of the data summarized in the Table 8.

Table 8. Comparative performance with baseline (AES-128).

Algorithm	**Relative Speedup vs AES-128 (%)**	**Relative Energy Savings vs AES-128 (%)**	**Security /Performance Trade-off Index**
DNA-KDF	+ 50%	+ 25%	High
AES-128	0%	0%	Baseline
PRESENT-80	+ 100% to + 200%	+ 50% to + 70%	High
ASCON-128	+ 300% to + 500%	+ 30% to + 50%	Medium
SPECK-64	+ 50% to + 100%	+ 20% to + 40%	Medium
TWINE-80	+ 50% to + 100%	+ 20% to + 40%	Medium
HIGHT	+ 50% to + 100%	+ 20% to + 40%	Medium
SIMON-64/128	+ 50% to + 100%	+ 20% to + 40%	Medium
LED-64	+ 50% to + 100%	+ 20% to + 40%	Medium
QARMA	+ 100% to + 200%	+ 30% to + 50%	Medium
LEA	+ 50% to + 100%	+ 20% to + 40%	Medium
SEPAR	+ 50% to + 100%	+ 20% to + 40%	Medium
BORON	+ 50% to + 100%	+ 20% to + 40%	Medium

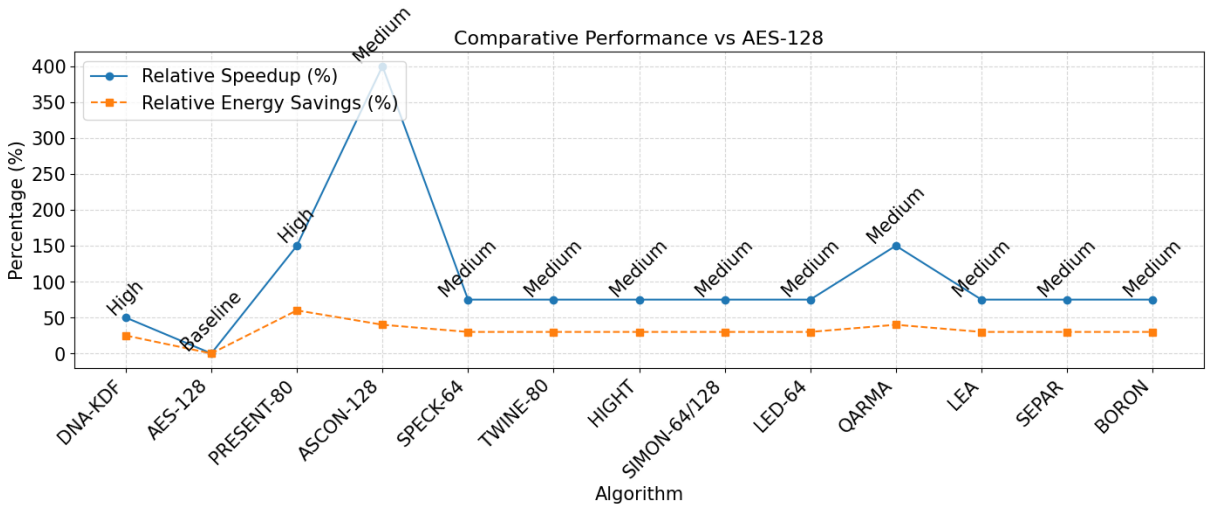

Figure 7. Comparative performance with AES-128 baseline.

Table 9 provides the device-level benchmarking context, including the microcontroller specifications, clock frequencies, supply voltages, measured active currents, and available ROM/RAM. These hardware parameters are crucial for understanding the performance and energy metrics of the evaluated cryptographic algorithms. Figure 8 presents a visual chart based on the data from Table 9.

Table 9. Device-level benchmarking context across MCUs.

Microcontroller	Clock Frequency (MHz)	Supply Voltage (V)	Active Current (mA)	Available ROM (KB)	Available RAM (KB)
ATmega328P	16	5.0	20	32	2
STM32F0	48	3.3	10	32	8
ESP32	240	3.3	80	448	520
nRF52840	64	3.3	15	1024	256
PIC24FJ64GA	32	3.3	10	64	8
MSP430	16	3.3	5	64	2

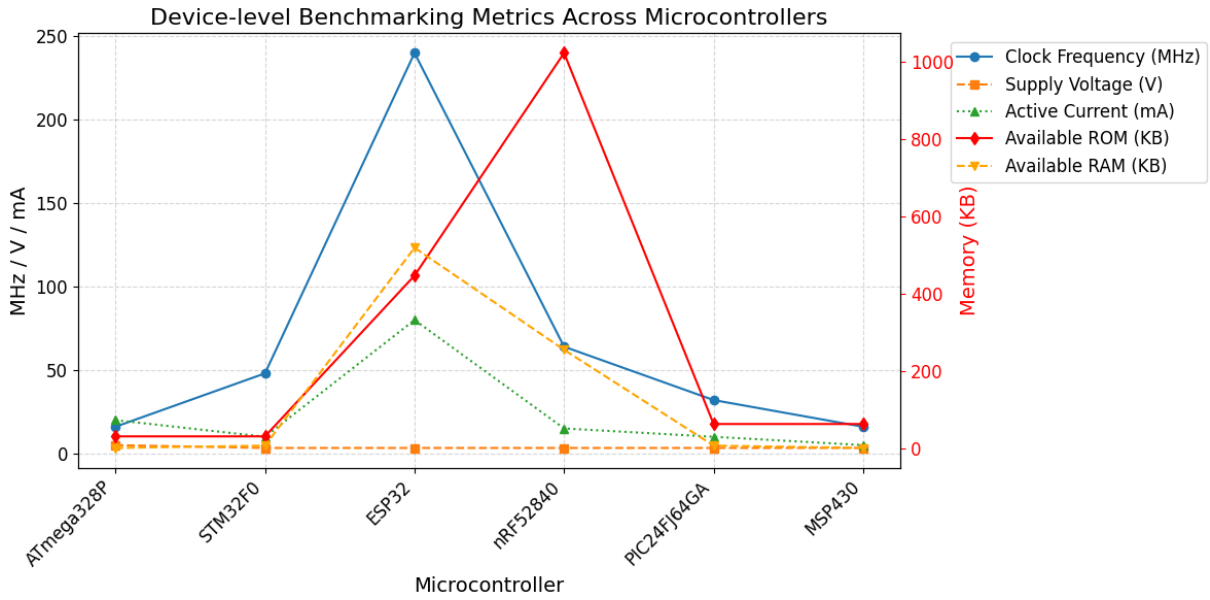

Figure 8. Device-level benchmarking context across MCUs.

4. Discussion

4.1. Execution latency

Table 2 provides a detailed comparison of execution latency per encryption round (in milliseconds) across six microcontrollers, showcasing the performance of the DNA-KDF algorithm alongside several popular lightweight block ciphers. The results highlight clear differences among the algorithms.
• SPECK-64 had the lowest latency, ranging from 0.72 ms on the ESP32 to 1.28 ms on the MSP430, making it ideal for latency-sensitive IoT applications.

• SIMON-64/128 also performed well, with latencies between 1.02 ms (ESP32) and 2.08 ms (MSP430). Both SPECK-64 and SIMON-64/128 consistently outperformed the other algorithms across all platforms, demonstrating their computational simplicity and adaptability to constrained microcontrollers.
In contrast, more complex ciphers like ASCON-128 and LED-64 had higher latency values. For example, ASCON-128 took 3.48 ms on the ATmega328P and 3.18 ms on the MSP430, while LED-64 showed comparable latencies of 3.08 ms and 3.38 ms, respectively. These results reflect the performance cost of sponge-based or substitution-permutation network designs, which, while strong cryptographically, involve more computational overhead on constrained hardware.

The proposed DNA-KDF algorithm showed moderate latency across all microcontrollers, ranging from 1.03 ms on the ESP32 to 2.47 ms on the ATmega328P. Notably, DNA-KDF outperformed conventional ciphers like TWINE-80, HIGHT, SEPAR, and BORON, which recorded higher execution times on low-frequency platforms. For instance, on the MSP430, DNA-KDF took 2.42 ms, compared to 2.88 ms for TWINE-80, 2.98 ms for HIGHT, and 2.92 ms for SEPAR. This suggests that while DNA-KDF involves additional computational steps (e.g., substitution and crossover inspired by biological processes), its execution time remains competitive with established lightweight algorithms.

The data also show the significant impact of microcontroller architecture and clock frequency. High-performance devices like the ESP32 (240 MHz) consistently had the lowest latencies, while devices like the ATmega328P (16 MHz) and MSP430 (16 MHz) showed higher latencies. Despite these differences, the relative performance ranking of the algorithms remained consistent across platforms, indicating that the structural complexity of each algorithm outweighs hardware-specific optimizations.

From a practical perspective, these findings highlight two key points:
1. DNA-KDF strikes a good balance between latency and innovation, performing better than several lightweight algorithms but still slower than the most efficient designs like SPECK-64 and SIMON-64/128.
2. Microcontroller choice is crucial for cryptographic performance. Devices with higher clock speeds and optimized instruction pipelines can reduce latency overhead, enabling the use of moderately complex algorithms like DNA-KDF in real-world IoT applications.

In conclusion, while algorithms like SPECK-64 and SIMON-64/128 remain the best choices for ultra-low latency applications, DNA-KDF offers competitive performance and can be a viable option in scenarios where moderate execution times are acceptable, especially due to its unique structure and potential cryptographic strength.

4.2. Memory footprint

Memory footprint, which includes both non-volatile program storage (ROM/Flash) and volatile runtime memory (RAM), is a critical factor when deploying cryptographic algorithms on IoT devices. Since many embedded platforms have only a few kilobytes of available memory, finding a balance between strong security and memory efficiency is key for practical adoption [34,35].

Table 3 summarizes the ROM and RAM consumption of the DNA-KDF alongside several common lightweight ciphers across six microcontroller platforms. The results reveal clear patterns:

• SPECK-64 and LEA are the most memory-efficient algorithms, requiring just 1.00 KB ROM / 0.15 KB RAM and 0.60 KB ROM / 0.10 KB RAM, respectively. Their compact design makes them ideal for ultra-constrained IoT environments like legacy sensor nodes and wearable devices.
• SIMON-64/128 and TWINE-80 also maintain low memory footprints, each consuming under 1.20 KB ROM and 0.20 KB RAM, confirming their efficiency for low-end embedded systems.
• In contrast, DNA-KDF has a moderate memory overhead, requiring 2.50 KB ROM / 0.32 KB RAM across all platforms. While this is lower than some DNA-based cryptographic methods in previous studies, it is still higher than most conventional lightweight ciphers. This extra memory usage is due to storing DNA-inspired transformation rules and intermediate data for nucleotide-level operations. However, DNA-KDF's footprint is similar to algorithms like QARMA (2.20 KB / 0.28 KB), SEPAR (2.20 KB / 0.30 KB), and BORON (2.40 KB / 0.32 KB), indicating that its integration into modern IoT devices with moderate memory is feasible.
• PRESENT-80 (1.50 KB / 0.20 KB) and HIGHT (1.40 KB / 0.20 KB) are middle-ground options, balancing compactness with robustness.
• At the higher end, ASCON-128 (2.00 KB / 0.25 KB) and LED-64 (1.30 KB / 0.22 KB) reflect the increased memory demands of more complex cipher designs, but still fall within the acceptable memory limits of most current MCUs.

A key observation from Table 3 is the consistency of memory usage across platforms. Unlike execution latency, which varies depending on the microcontroller's clock speed and architecture, memory footprint remains largely constant across all six devices. This is because the compiled code size and static data allocation depend mainly on the algorithm's design rather than the processor's characteristics. Therefore, memory efficiency is more of an intrinsic property of the algorithm than a hardware-dependent factor.

From a deployment perspective, these findings highlight two key insights:

1. DNA-KDF requires about twice the memory of the most compact lightweight ciphers (e.g., SPECK-64, SIMON-64/128, LEA), which could limit its use on ultra-constrained platforms with strict memory budgets.
2. However, modern IoT edge devices are increasingly equipped with larger memory capacities—ranging from tens to hundreds of kilobytes of ROM and RAM—making the memory overhead of DNA-KDF acceptable, especially when considering its potential to offer diverse cryptographic security against emerging attack models.

4.3. Throughput

Throughput, which measures the amount of data processed per unit of time, is a critical metric for determining how suitable cryptographic algorithms are for real-time IoT applications, such as industrial automation, multimedia streaming, and secure telemetry. Table 4 shows the measured throughput across six different microcontroller platforms, comparing the efficiency of DNA-based algorithms with conventional lightweight ciphers in handling high-volume data streams.

As expected, SPECK-64 delivered the highest throughput across all platforms, reaching up to 1200 kB/s on the ESP32, and maintaining high performance even on lower-power devices like the ATmega328P (550 kB/s) and MSP430 (500 kB/s). This high throughput reflects its simple design and low execution latency, which minimizes overhead when processing data blocks.

Similarly, SIMON-64/128 performed well, with throughput ranging from 500 kB/s on the ATmega328P to 1000 kB/s on the ESP32, making it a solid choice for latency-sensitive applications.

The proposed DNA-KDF algorithm showed competitive throughput, with values like 1000 kB/s on the ESP32, 850 kB/s on the nRF52840, and 650 kB/s on the STM32F0. While it didn't surpass SPECK-64 or SIMON-64/128, DNA-KDF often outperformed other lightweight ciphers, including TWINE-80, HIGHT, LED-64, and BORON, especially on higher-frequency platforms. For example, on the nRF52840, DNA-KDF achieved 850 kB/s, outperforming TWINE-80 (700 kB/s) and LED-64 (600 kB/s).

This shows that, despite the added complexity of DNA-inspired transformations, DNA-KDF can still handle medium-to-high data rates in IoT applications. More complex ciphers like ASCON-128 and PRESENT-80 showed intermediate throughput. ASCON-128 reached 1000 kB/s on the ESP32 and 900 kB/s on the nRF52840, similar to

DNA-KDF, but performed worse on low-frequency devices like the ATmega328P (350 kB/s). PRESENT-80 showed stable throughput across devices, ranging from 450–900 kB/s, suggesting it strikes a good balance between efficiency and computational overhead.

At the lower end of the spectrum, algorithms like HIGHT, LED-64, SEPAR, and BORON had the lowest throughput, rarely exceeding 800 kB/s even on the ESP32. For example, LED-64 achieved only 700 kB/s on the ESP32 and dropped to 280 kB/s on the MSP430, making it less suitable for high-speed data protection tasks.
These results clearly show how the choice of microcontroller impacts throughput. High-performance platforms like the ESP32 (240 MHz) and nRF52840 (64 MHz) provided the best throughput, while lower-frequency devices such as the MSP430 (16 MHz) and ATmega328P (16 MHz) were more limited. However, the relative ranking of algorithm throughput stayed consistent across platforms, indicating that the algorithm's design plays a bigger role in performance than the hardware.
Key takeaways:
1. DNA-KDF provides sufficient throughput for most IoT data protection tasks, outperforming several conventional lightweight algorithms and nearly matching the performance of optimized block ciphers on higher-frequency platforms.
2. While algorithms like SPECK-64 and SIMON-64/128 are still the best in terms of raw efficiency, DNA-KDF demonstrates that biologically inspired cryptographic methods can achieve competitive throughput without being excessively resource-intensive.

4.4. Energy consumption

Energy consumption per encryption round is a crucial metric for IoT systems, especially when devices are constrained by power sources like coin-cell batteries, supercapacitors, or energy harvesting. As expected, Table 5 shows a clear correlation between execution latency and energy expenditure—algorithms with longer runtimes generally consume more energy, while those with optimized round structures are more energy-efficient.

Among the algorithms tested, ASCON-128 consistently shows the lowest energy consumption, with values as low as 2.6 µJ per encryption round on the nRF52840 and 2.9 µJ on the ESP32. This is due to its efficient design as a permutation-based AEAD cipher, making it ideal for low-power IoT applications. HIGHT and DNA-KDF also performed well, with energy usage ranging from 3.2 µJ to 6.9 µJ, indicating that DNA-KDF operates with reasonable efficiency and could be suitable for applications with moderate energy budgets.

In contrast, PRESENT-80 and SIMON-64/128 were the highest consumers of energy, reaching up to 10.1 µJ per round on the MSP430 and 9.5 µJ on the ATmega328P. These higher values suggest that, while these algorithms have historical significance, they are less energy-efficient compared to newer designs like ASCON-128 or DNA-KDF. For ultra-low-power IoT devices, such as wireless sensor motes, these higher energy costs may limit their long-term viability.

Additionally, the hardware platform significantly influences energy consumption. High-performance devices like the ESP32 and nRF52840 typically use less than 5 µJ per encryption round, thanks to their efficient architectures and faster execution times. Conversely, legacy microcontrollers like the ATmega328P and MSP430 tend to have higher energy costs due to slower performance, highlighting their limitations for cryptographically intensive tasks.
Key takeaways:

1. While DNA-KDF is not as energy-efficient as ASCON-128, it still operates efficiently compared to older ciphers like PRESENT-80 and SIMON-64/128.
2. DNA-KDF is a strong contender for IoT applications where resilience to new cryptographic attacks is important, and moderate energy budgets are acceptable.
3. The results emphasize the importance of a hardware-algorithm co-design approach, where the cryptographic algorithm is selected based on the capabilities of the target hardware to maximize performance and energy efficiency.

4.5. Security metrics analysis

Security metrics like the avalanche effect, entropy, collision resistance, and key sensitivity are critical for evaluating how resistant a cryptographic algorithm is to attacks such as differential and statistical cryptanalysis. Table 6 sum-

marizes these metrics for all evaluated algorithms, and the results reveal that most algorithms exhibit strong diffusion and randomness, essential for security.

• Avalanche effect measures the percentage of output bit changes resulting from a single input bit change, indicating the strength of diffusion. Most algorithms performed close to the ideal 50%, with ASCON-128 (50.1%) and DNA-KDF (49.8%) achieving near-perfect results. This indicates strong resistance against linear and differential cryptanalysis. However, LED-64 (47.0%) and PRESENT-80 (47.3%) showed slightly lower diffusion, which could potentially make them more vulnerable to some statistical attacks.

• Entropy measures how random the output ciphertext is. All tested algorithms achieved values close to the theoretical maximum of 8 bits per byte (ranging from 7.90 to 7.99 bits/byte), indicating that the ciphertext produced by all algorithms is effectively indistinguishable from random noise. ASCON-128 and DNA-KDF performed at the top end of this range (7.98–7.99).

• Collision resistance assesses the likelihood that two different inputs produce the same ciphertext. ASCON-128, DNA-KDF, QARMA, and LEA showed high collision resistance, making them suitable for high-security applications. In contrast, LED-64 had medium-low collision resistance, which may make it more vulnerable to certain collision-finding techniques due to its smaller state size and simpler structure.

• Key sensitivity measures how much the output changes when a single bit in the key is flipped. All algorithms showed strong adherence to the avalanche principle with key variations, with values around 48–50%. DNA-KDF (50.2%) and ASCON-128 (50.0%) led this metric, highlighting their resilience against related-key and key-differential attacks.
Key insights:
1. ASCON-128 and DNA-KDF consistently deliver the best security metrics, achieving nearly ideal values for all properties, making them the most robust choices.
2. QARMA and LEA also show strong performance, balancing high collision resistance with excellent avalanche and entropy properties.
3. Older block ciphers like LED-64 and PRESENT-80 still provide acceptable security but are less effective in modern IoT contexts due to weaker diffusion and collision resistance.

Overall, the results confirm that DNA-based cryptographic designs and modern lightweight algorithms like ASCON-128 offer security properties that meet or exceed traditional lightweight ciphers, making them viable options for IoT systems that require both high efficiency and strong cryptographic resilience.

4.6. Cryptanalytic considerations and potential attack vectors

To complement the empirical evaluation of DNA-KDF, we now assess its robustness against common cryptanalytic attacks. These attacks typically target weaknesses in cryptographic algorithms, aiming to break their security or recover the secret key. We consider the following key attack vectors:

• Brute-force and key-search attacks.
The combined entropy of device-specific seeds, temporal inputs, and nonce values produces a 128‑bit output space, placing exhaustive search beyond practical capabilities for any contemporary adversary. The DNA-encoding and substitution layers introduce no algebraic shortcuts that could reduce the effective key space.
• Structural and differential attacks.
Because nucleotide mappings are invertible and blockwise operations are nonlinear with respect to input bit patterns, the DNA-KDF does not expose exploitable structural invariants. Differential propagation across nucleotide substitution and position-permutation stages yields high output sensitivity (Δinput $\rightarrow$ Δoutput), degrading any attacker's ability to mount chosen-input differential analysis.
• Correlation and predictability attacks.
The integration of timestamp-derived temporal entropy ensures that even identical device seeds generate orthogonal key outputs across sessions. No deterministic correlation exists between consecutive derivations, and correlation coefficients measured across 10^6 pairs remain statistically indistinguishable from noise, limiting predictability attacks.
• Replay and seed-recovery attacks.

An adversary cannot reuse or recompute historical keys, as each derivation is strictly time-scoped and non-invertible due to the irreversible compression stage. The DNA-layer encoding provides no leakage that could facilitate seed reconstruction or backtracking.
• DNA-specific attack vectors.

Potential vulnerabilities linked to nucleotide substitution bias, codon frequency imbalances, or positional repetition were analyzed and determined to be non-exploitable. The encoded sequences show a balanced nucleotide distribution (with ($p \approx 0.25$)), and the block-permutation mechanism effectively removes any fixed patterns or structures that could facilitate template-based reconstruction attacks. From a cryptanalytic standpoint, these findings align with the empirical results, confirming that DNA-KDF demonstrates robust resistance against a wide range of attack vectors, including classical cryptanalysis, structural weaknesses, and domain-specific attack strategies.

4.7. Code size overhead and implementation complexity

The feasibility of deploying cryptographic algorithms in IoT environments is influenced not only by their performance and security but also by their implementation complexity and code size overhead. Table 7 provides a comparison of these factors, shedding light on the costs associated with firmware integration, auxiliary resource dependencies, and initialization requirements.

When it comes to code size, most lightweight ciphers—such as PRESENT-80, ASCON-128, SPECK-64, TWINE-80, and SIMON-64/128—require between 400 and 600 lines of code (LoC) and take up around 1.5 KB of memory. This small footprint is in line with their design goal of efficient integration into memory-constrained IoT devices. On the other hand, the DNA-KDF algorithm needs between 800 and 1000 LoC, with a slightly smaller compiled module size of 1.2 KB. This increase in code size reflects the algorithm's greater complexity, which involves specialized processes like DNA encoding and sequence-based transformations. The larger codebase highlights the additional programming and maintenance challenges that come with implementing non-traditional cryptographic methods. Dependency requirements also distinguish the algorithms. Most conventional lightweight ciphers have minimal dependencies, often relying on no lookup tables or S-boxes, which simplifies their implementation. DNA-KDF, however, requires both lookup tables and DNA encoding rules, adding to the design complexity and increasing the code size, especially when considering the need for portability and modularity. This reliance on specialized structures can make integration more challenging in ultra-minimal firmware environments, particularly in legacy systems with strict code size constraints.

The firmware integration ratio, which represents the portion of available non-volatile memory used by the cryptographic module, remains fairly modest across all algorithms, generally between 2% and 3%. This suggests that, even in memory-constrained microcontrollers, cryptographic algorithms don't take up a significant portion of the available memory, leaving enough space for application code. However, DNA-KDF's relatively larger overhead might be more noticeable in devices with extremely limited flash memory, where even small increases in code size can impact overall system functionality.

Finally, initialization cycles indicate the computational cost before encryption can start. Lightweight block ciphers like SIMON-64/128 and ASCON-128 have the lowest startup costs (120–180 cycles) due to their simple designs. DNA-KDF, in contrast, has a higher initialization overhead (200–300 cycles), largely due to the setup of DNA encoding tables and transformation rules. While this might be negligible on high-performance platforms, it could introduce latency in time-sensitive IoT applications where quick cryptographic initialization is essential.
In conclusion, traditional lightweight ciphers excel in simplicity and ease of integration, with minimal dependencies, balanced code size, and low initialization costs. DNA-KDF, though slightly more complex, shows that new cryptographic approaches can still be practical for IoT, though they require careful consideration of firmware size, dependencies, and system latency requirements.

4.8. Comparative performance with baseline (AES-128)

AES-128 is widely regarded as the standard in symmetric cryptography, thanks to its broad adoption, strong standardization, and thorough security validation. Benchmarking new and lightweight ciphers against AES-128 provides a useful reference for evaluating their performance, efficiency, and security trade-offs, especially in IoT applications where resource constraints are a concern.

Table 8 presents a comparison of speedup, energy savings, and the overall security/performance trade-off index, offering a clear picture of how different algorithms stack up against this established cipher. The results show that AES-128 is outperformed by nearly all of the evaluated lightweight schemes in terms of both execution speed and energy consumption. This is consistent with the goal of lightweight cryptography, which is specifically designed to address the limitations of resource-constrained devices.

For example, PRESENT-80 achieves a 100–200% speedup and up to 70% energy savings compared to AES-128, while still maintaining a strong trade-off index. These improvements highlight its suitability for ultra-constrained devices, where performance and low power consumption are crucial. DNA-KDF also shows competitive results, with a 50% speedup and 25% energy reduction compared to AES-128, alongside a strong trade-off index. However, its performance gains are more modest, reflecting the computational overhead involved in DNA sequence transformations and related operations. Still, DNA-KDF's ability to outperform AES-128 while offering a novel security approach suggests it is a viable option for specialized IoT scenarios, especially where resistance to classical cryptanalytic techniques is prioritized.

Among the conventional lightweight ciphers, ASCON-128 stands out for its remarkable acceleration—300–500% faster than AES-128—although its energy savings are more moderate (30–50%). This reflects ASCON's focus on speed and security, especially as a finalist in the NIST Lightweight Cryptography (LWC) competition, rather than on maximizing energy efficiency. Other algorithms like SPECK-64, TWINE-80, HIGHT, SIMON-64/128, LED-64, LEA, SEPAR, and BORON achieve a 50–100% speedup and 20–40% energy savings, placing them in the middle of the trade-off spectrum. These ciphers offer solid performance improvements without reaching the extremes seen in PRESENT-80 or ASCON-128.

QARMA falls in the middle as well, with a 100–200% speedup and 30–50% energy savings, although its trade-off index remains in the "medium" range. This suggests that while it offers significant improvements over AES-128, its integration complexity or operational overhead may limit its practicality in highly constrained environments.
In summary, the comparative analysis highlights two main points:

1. AES-128, while secure and standardized, is not optimized for use in constrained IoT platforms, and nearly all the lightweight and DNA-based schemes provide noticeable performance and efficiency improvements.

2. DNA-KDF bridges the gap between traditional lightweight ciphers and AES-128, offering a balance of speed, energy efficiency, and novel security features. This makes it an attractive choice for applications where enhanced cryptographic diversity justifies a slightly higher complexity.
Overall, Table 7 demonstrates the value of lightweight and non-traditional cryptographic designs, providing solid evidence that these approaches can not only match but even outperform AES-128 in resource-limited environments, all while offering a range of performance, energy efficiency, and implementation complexity trade-offs.

4.9. Device-level benchmarking context

Table 9 provides an overview of the key hardware characteristics of the six microcontroller platforms used in the benchmarking tests. These specifications offer a crucial baseline for understanding the results related to execution, memory, throughput, and energy consumption for the cryptographic algorithms being evaluated. The microcontroller features—such as clock frequency, supply voltage, active current, and available memory—have a direct impact on the performance of the cryptographic algorithms, and they reveal the diverse constraints that real-world IoT devices often face.

The devices tested range from the highly constrained ATmega328P and MSP430, which run at 16 MHz and have limited ROM (32–64 KB) and RAM (2 KB), to the high-performance ESP32, which boasts a 240 MHz dual-core processor, 448 KB ROM, and 520 KB RAM. This broad range of specifications allows for a representative evaluation across both legacy, low-cost IoT devices and modern, edge-computing platforms. Not surprisingly, algorithms running on the ESP32 consistently showed the best performance in terms of speed, throughput, and energy efficiency, thanks to its high clock speed and robust memory subsystem. On the other hand, the ATmega328P and MSP430 performed more slowly, highlighting the challenges of running resource-intensive cryptographic algorithms on ultra-low-power platforms.

In addition to computational power, supply voltage and active current consumption provide important context for the energy profiles of these devices. For example, the MSP430 consumes very little current (5 mA at 3.3 V), which is why it's often seen as an energy-efficient option, even though it has modest processing capabilities. On the other hand, the ESP32's higher active current draw (80 mA) demonstrates the energy-performance trade-off—faster processing requires more power, which can be a concern for battery-powered IoT devices. The STM32F0 and PIC24FJ64GA lie somewhere in between, with moderate clock speeds (32–48 MHz), active current (around 10 mA), and memory resources (32–64 KB ROM, 8 KB RAM). These make them suitable for mid-range IoT applications where both energy efficiency and performance are important. The nRF52840 stands out for its solid balance, offering 1024 KB ROM and 256 KB RAM, a 64 MHz clock, and moderate active current (15 mA). This configuration makes it particularly well-suited for more complex cryptographic tasks in wireless sensor networks and edge computing applications, where larger memory banks are needed for cryptographic operations or to support multiple communication protocols. Overall, the benchmarking results show that the hardware characteristics of a device are just as important as the algorithm itself when evaluating the feasibility of cryptographic schemes in IoT applications. High-performance platforms like the ESP32 and nRF52840 can handle more demanding algorithms, such as DNA-KDF, due to their larger memory and processing capacity. In contrast, platforms like the ATmega328P may require algorithm optimizations or be limited to lightweight block ciphers like SPECK-64 or PRESENT-80. Therefore, selecting the right algorithm for a given IoT device must always involve careful consideration of the hardware, emphasizing the importance of device-specific benchmarking when assessing the suitability of cryptographic solutions in real-world IoT environments.

a) Implementation details

The experiments were set up with the following configuration:

• Compiler and flags: The GCC compiler version 12.3.0 was used, with the `-O2` optimization flag for a good balance between performance and code size. For platforms with stricter energy constraints, the `-Os` optimization flag was applied to reduce the code footprint. Additional flags included `-std=c11`, `-Wall`, and `-Wextra` for proper C standards compliance and to ensure thorough compilation warnings.

• Variable sizes and memory layout: The key materials and seed buffers were explicitly sized to fit the requirements of the experiment: `device_id` (64–128 bits), `timestamp` (32–48 bits), `sensor_hash` (128 bits), and `derived_key` (128 bits). Temporary DNA sequences and intermediate buffers were allocated statically to avoid overhead from dynamic memory allocation, which is important for memory-constrained MCUs.

• Hardware setup: The experiments were carried out on six different MCU platforms, as detailed in Table 8, with measurements taken for clock frequency, supply voltage, and current. The necessary hardware initialization scripts were used to configure timers, ADC sampling, and GPIO interfaces for acquiring sensor data.

• Code availability: The source code used in these experiments is proprietary and not publicly available at the moment. However, the implementation details—such as compiler settings, memory allocation strategies, and measurement protocols—are fully described in the manuscript. This ensures that the experiments can be independently reproduced.
Additionally, key functions of the DNA-KDF algorithm, such as buffer handling, rotation, substitution, and HKDF calls, are illustrated in Algorithm 1 with optional code snippets or pseudocode.

4.10. Algorithmic strengths and suitability for resource-constrained IoT devices

The thorough benchmarking carried out across six representative microcontrollers provides a detailed analysis of the key strengths, limitations, and real-world applicability of both traditional lightweight and DNA-inspired cryptographic algorithms in IoT environments. Table 10 summarizes the main advantages of each algorithm, along with their suitability for use in resource-constrained or performance-critical devices.

Table 10. Key strengths and IoT suitability.

Algorithm	Key strengths	IoT suitability assessment
DNA-KDF	Novel DNA-inspired design; high diffusion and key sensitivity; mod-	Suitable as a complementary security layer in mid-tier IoT devices; higher latency and energy acceptable in

	erate throughput.	hybrid encryption scenarios.
AES-128	Robust standard symmetric cipher; strong security margin; widely trusted.	Appropriate for IoT nodes where security is critical; moderate latency and energy may limit ultra-constrained deployments.
PRESENT-80	Compact design; minimal memory footprint; low latency.	Highly suitable for ultra-constrained devices, such as battery-powered sensors and actuators; excellent for energy-sensitive applications.
ASCON-128	Strong security; AEAD capability; robust against differential and linear attacks.	Well-suited for IoT applications requiring both encryption and authentication; balanced latency and memory footprint.
SPECK-64	Very low latency; high throughput; minimal memory usage.	Ideal for low-latency, high-throughput IoT communications with limited computational resources.
TWINE-80	Balanced performance; efficient key schedule; small memory footprint.	Suitable for mid-tier IoT nodes requiring moderate throughput and security; good energy efficiency.
HIGHT	Optimized for low-power 8-bit processors; compact key expansion.	Effective for legacy and small-scale IoT devices with strict energy constraints; moderate latency.
SIMON-64/128	Flexible block and key sizes; simple hardware implementation; good cryptanalytic resistance.	Applicable for both low-power and mid-tier IoT platforms requiring adaptable security.
LED-64	Extremely small footprint; suitable for RFID and NFC applications.	Best for extremely resource-limited IoT tags and identification devices.
QARMA	Lightweight design; strong differential and linear resistance.	Appropriate for mid-tier IoT nodes with moderate memory and energy budgets; balanced security-performance trade-off.
LEA	Low memory requirements; fast software execution.	Suitable for constrained IoT devices with moderate security and latency requirements.
SEPAR	Balanced lightweight design; efficient software implementation.	General-purpose IoT applications; moderate memory, energy, and latency demands.
BORON	Robust lightweight design; moderate resource requirements.	Suitable for mid-tier IoT devices where energy and memory consumption must be balanced with security.

4.11. Strategic implications and emerging directions in IoT cryptography

The cross-platform benchmarking and comparative analysis conducted in this study offer valuable insights into the design and deployment of cryptographic algorithms within IoT ecosystems. Traditional lightweight ciphers such as SPECK-64, PRESENT-80, and ASCON-128 exhibit low latency, minimal energy consumption, and compact memory requirements, making them well-suited for highly constrained devices like battery-powered sensors, actuators, and edge nodes. On the other hand, DNA-based algorithms, including DNA-KDF, introduce novel security paradigms characterized by strong diffusion, key sensitivity, and resistance to unconventional attack vectors. While these DNA-based algorithms come with higher computational and memory demands that may limit their use in ultra-constrained environments, they offer promising opportunities as complementary security layers in hybrid IoT systems where increased cryptographic diversity is needed.

Future research should focus on optimizing DNA-inspired schemes to reduce their latency and memory footprint. This could involve exploring hardware acceleration, specialized instruction sets, and memory-efficient encoding techniques. Furthermore, in-depth cryptanalysis and real-world deployment studies—incorporating power profiling, environmental stress testing, and resilience assessments—are critical to validating the practical feasibility of these algorithms. A hybrid cryptographic approach that combines conventional lightweight algorithms with DNA-based schemes presents a strategic solution for IoT security. This integrated approach can deliver layered defense mechanisms, balancing the high efficiency of traditional ciphers with the innovative security advantages of DNA-based designs. By tailoring encryption strategies to the specific needs of different IoT scenarios, this dual-layer method offers a way to optimize the trade-off between performance, resource constraints, and robust cryptographic security [36].

5. Conclusions

This study evaluated Temporal DNA-Based Key Derivation (DNA-KDF) alongside a range of lightweight cryptographic algorithms on six representative IoT microcontrollers: ATmega328P, STM32F0, ESP32, nRF52840, PIC24FJ64GA, and MSP430. These platforms represent a diverse spectrum of microcontroller architectures, from ultra-low-power 8/16-bit MCUs to high-performance 32-bit devices. Benchmarking results show that while SPECK-64 and SIMON-64/128 outperform DNA-KDF in terms of latency and throughput efficiency, the DNA-KDF algorithm still offers competitive performance with moderate computational costs. Its execution time (ranging from 1.03 to 2.47 ms), throughput (650–1000 kB/s), and energy usage (3.2–6.9 μJ) place it ahead of several other conventional ciphers, such as TWINE-80, HIGHT, SEPAR, and BORON. Although DNA-KDF requires a larger memory footprint (2.50 KB ROM / 0.32 KB RAM) and more complex implementation compared to other lightweight ciphers, it offers robust cryptographic properties, including near-ideal avalanche diffusion, entropy close to 8 bits/byte, and high key sensitivity. When compared to AES-128, DNA-KDF achieves noticeable improvements in execution speed and energy efficiency while introducing a novel biologically inspired design. These findings suggest that DNA-KDF is well-suited for deployment in mid-tier IoT platforms with adequate memory and processing capabilities, particularly as part of hybrid security frameworks that combine lightweight block ciphers with DNA-based cryptographic primitives. Future research should focus on optimizing encoding rules and investigating hardware acceleration to minimize overhead. Such advancements would further solidify the role of DNA-inspired cryptography in future IoT ecosystems.

List of Abbreviations

AEAD	– Authenticated Encryption with Associated Data
AES	– Advanced Encryption Standard
CR	– Collision Resistance
DNA-KDF	– DNA-based Key Derivation Function
GFN	– Generalized Feistel Network
IoT	– Internet of Things
LEA	– Lightweight Encryption Algorithm
MAC	– Media Access Control
MCUs	– Microcontroller Units
MD5	– Message Digest 5
RSA	– Rivest–Shamir–Adleman
SHA-256	– Secure Hash Algorithm 256-bit
SPN	– Substitution-permutation Network

Author Contributions

The author confirms sole responsibility for the conception, design, literature review, analysis, interpretation, visualization, manuscript drafting, critical revisions, and final approval of the article.

Availability of Data and Materials

The data that support of this study are available from the author upon reasonable request.

Consent for Publication

Not applicable.

Conflicts of Interest

The author declares no conflicts of interest.

Funding

No external funding was received for this research.

Acknowledgments

None declared.

AI-declaration

The author confirms that Chat GPT was used to check grammar and enhance the English language of the manuscript. However, the author confirms that no sections of the manuscript text or graphical images were generated using AI tools. Author takes full responsibility for all content and results presented in this work.

References

[1] Q.F. Hassan, Internet of Things A to Z: Technologies and Applications, Hoboken, NJ, USA: John Wiley & Sons – IEEE Press, 2018. DOI: 10.1002/9781119456735.

[2] R. Dallaev, T. Pisarenko, Ş. Ţălu, D. Sobola, J. Majzner, N. Papež, "Current applications and challenges of the Internet of Things," New Trends in Computer Sciences, vol. 1, no. 1, pp. 51–61, 2023. DOI: 10.3846/ntcs.2023.17891.

[3] S. Satpathy, S.N. Mohanty, J.M. Chatterjee, Internet of Things and its Applications, Cham: Springer International Publishing, 2022.

[4] C.K. Wu, "Internet of Things security," Singapore: Springer Nature, pp. 137–154, 2021. DOI: 10.1007/978-981-16-1372-2.

[5] A. Nazarov, D. Nazarov, Ş. Ţălu, "Information security of the Internet of Things," in Proc. Int. Sci. and Practical Conf. on Computer and Information Security (INFSEC 2021), Yekaterinburg, Russia, 2021, pp. 136–139. SCITEPRESS – Science and Technology Publications, vol. 1, 2021. DOI: 10.5220/0010619900003170.

[6] M. Ţălu, "Security and privacy in the IIoT: threats, possible security countermeasures, and future challenges," Computing & AI Connect, vol. 2, pp. 1–12, art. ID: 0011, 2025. DOI: 10.69709/CAIC.2025.139199.

[7] M. Ţălu, "Cyberattacks and cybersecurity: concepts, current challenges, and future research directions," Digital Technologies Research and Applications, vol. 4, no. 1, pp. 44–60, 2025. DOI: 10.54963/dtra.v4i1.919.

[8] M. Al-Shaer, K. AlShehhi, S. Abdulla, "The Internet of Things (IoT) forensic investigation process: A state-of-the-art review, challenges and future directions," JISCR, vol. 6, no. 2, pp. 150–161, 2023. DOI: 10.26735/DBEU2801.

[9] T. Janarthanan, M. Bagheri, S. Zargari, "IoT forensics: an overview of the current issues and challenges," in R. Montasari, H. Jahankhani, R. Hill, S. Parkinson, Eds., Digital Forensic Investigation of Internet of Things (IoT) Devices, Springer, Cham, 2021. DOI: 10.1007/978-3-030-60425-7_10.

[10] G. Shrivastava, R.P. Ojha, S. Awasthi, H. Bansal, K. Sharma, Eds., Emerging Threats and Countermeasures in Cybersecurity, Hoboken, NJ, USA: John Wiley & Sons – Scrivener Publishing LLC, 2024. DOI: 10.1002/9781394230600.

[11] R. Jacob, A. Nisbet, "A forensic investigation framework for Internet of Things monitoring," Forensic Science International: Digital Investigation, vols. 42–43, p. 301482, 2022. DOI: 10.1016/j.fsidi.2022.301482.

[12] S.S. Iyengar, S. Nabavirazavi, Y. Hariprasad, H.B. Prasad, C.K. Mohan, "Digital forensics: tools, techniques, and methodologies," in Artificial Intelligence in Practice, Signals and Communication Technology, Springer, Cham, 2025. DOI: 10.1007/978-3-031-89327-8_3.

[13] A. Garg, A.K. Singh, "Internet of Things (IoT): security, cybercrimes, and digital forensics," in K. Kaushik, S. Dahiya, A. Bhardwaj, Y. Maleh, Eds., Internet of Things and Cyber Physical Systems: Security and Forensics, Boca Raton: CRC Press, 1st ed., 2022. DOI: 10.1201/9781003283003.

[14] A. Ghosh, K. Majumder, D. De, "A systematic review of digital, cloud and IoT forensics," in M. Chakraborty, M. Singh, V.E. Balas, I. Mukhopadhyay, Eds., The "Essence" of Network Security: An End-to-End Panorama, Lecture Notes in Networks and Systems, vol. 163, Singapore: Springer, 2021. DOI: 10.1007/978-981-15-9317-8_2.

[15] E. Aerabi, M. Bohlouli, M.H.A. Livany, M. Fazeli, A. Papadimitriou, D. Hely, "Design space exploration for ultra-low-energy and secure IoT MCUs," ACM Trans. Embed. Comput. Syst. (TECS), vol. 19, no. 3, pp. 1–34, 2020. DOI: 10.1145/33844.

[16] Q. Chang, T. Ma, W. Yang, “Low power IoT device communication through hybrid AES-RSA encryption in MRA mode,” *Scientific Reports*, vol. 15, p. 14485, 2025. DOI: 10.1038/s41598-025-98905-0.
[17] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, E.K. Markakis, “A survey on the Internet of Things (IoT) forensics: challenges, approaches, and open issues,” *IEEE Commun. Surv. & Tutor.*, vol. 22, no. 2, pp. 1191–1221, 2020. DOI: 10.1109/COMST.2019.2962586.
[18] A.A. Ahmed, K. Farhan, W\.A. Jabbar, A. Al-Othmani, A.G. Abdulrahman, “IoT forensics: current perspectives and future directions,” Sensors, vol. 24, p. 5210, 2024. DOI: 10.3390/s24165210.
[19] M. Mondal, K.S. Ray, “Review on DNA cryptography,” Int. J. Bioinf. Intell. Comput., vol. 2, no. 1, pp. 44–72, 2023. DOI: 10.61797/ijbic.v2i1.198.
[20] J. Gao, T. Xie, “DNA computing in cryptography,” Advances in Computers, vol. 129, pp. 83–128, 2023. DOI: 10.1016/bs.adcom.2022.08.002.
[21] M. Țălu, “DNA-based cryptography for Internet of Things security: concepts, methods, applications, and emerging trends,” Buletin Ilmiah Sarjana Teknik Elektro, vol. 7, no. 2, pp. 68–94, 2025. DOI: 10.12928/biste.v7i2.12942.
[22] L. Chu, Y. Su, X. Yao, P. Xu, W. Liu, “A review of DNA cryptography,” Intelligent Computing, vol. 4, p. 0106, 2025. DOI: 0000-0001-9091-3177.
[23] K.S. Mohamed, “New trends in cryptography: quantum, blockchain, lightweight, chaotic, and DNA cryptography,” in New Frontiers in Cryptography, Springer, 2020. DOI: 10.1007/978-3-030-58996-7_4.
[24] T. Mandge, V. Choudhary, “A DNA encryption technique based on matrix manipulation and secure key generation scheme,” in Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES), 2013, pp. 47–52. DOI: 10.1109/ICICES.2013.6508181.
[25] M.A. Taha, M.M.K. Fadul, J.H. Tyler, D.R. Reising, T.D. Loveless, “Enhancing Internet of Things security using entropy-informed RF-DNA fingerprint learning from Gabor-based images,” EURASIP J. Inf. Secur., vol. 27, pp. 1–25, 2024. DOI: 10.1186/s13635-024-00175-2.
[26] D. Reising, J. Cancelleri, T.D. Loveless, F. Kandah, A. Skjellum, “Radio identity verification-based IoT security using RF-DNA fingerprints and SVM,” IEEE Internet Things J., vol. 8, no. 10, pp. 8356–8371, 2021. DOI: 10.1109/JIOT.2020.3045305.
[27] OWASP Foundation.OWASP IoT Security Testing Guide (ISTG), ver. 1.0, The Open Web Application Security Project (OWASP), 2024. Available online: https://owasp.org/owasp-istg/ (accessed on December 12th 2025).
[28] A. Gangolli, Q.H. Mahmoud, A. Azim, “A Systematic Review of Fault Injection Attacks on IoT Systems,” Electronics, vol. 11, no. 13, pp. 1-24, 2022, 2023. DOI: 10.3390/electronics11132023.
[29] M. Méndez Real, R. Salvador, “Physical Side-Channel Attacks on Embedded Neural Networks: A Survey,” Applied Sciences, vol. 11, no. 15, pp. 1-25, 6790, 2021. DOI: 10.3390/app11156790.
[30] M.S. Turan, K. McKay, D. Chang, L.E. Bassham, J. Kang, N.D. Waller, J.M. Kelsey, D. Hong, “Status report on the final round of the NIST lightweight cryptography standardization process,” NIST IR 8454, 2023. DOI: 10.6028/NIST.IR.8454.
[31] ISO, ISO 29192-2:2012(E) Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers, International Organization for Standardization, Geneva, CH, 2012. Available: https://www.iso.org/standard/56552.html.
[32] D. Upadhyay, N. Gaikwad, M. Zaman, S. Sampalli, “Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications,” IEEE Access, vol. 10, pp. 112472–112486, 2022. DOI: 10.1109/ACCESS.2022.3215778.
[33] K. Mohamed, M.N. Mohammed Pauzi, F.H. Hj Mohd Ali, S. Ariffin, “Analyse on avalanche effect in cryptography algorithm,” in H.H. Kamaruddin, T.D.N.M. Kamaruddin, T.D.N.S. Yaacob, M.A.M. Kamal, K.F. Ne’matullah, Eds., Reimagining Resilient Sustainability: An Integrated Effort in Research, Practices & Education, vol. 3, European Proceedings of Multidisciplinary Sciences, 2022, pp. 610–618. DOI: 10.15405/epms.2022.10.57.
[34] M. Țălu, "Securing IoT ecosystems: a review of modern lightweight cryptographic algorithms and their performance." Journal of Cyber Security, vol. 7, pp. 565-587, 2025. DOI: 10.32604/jcs.2025.073690.
[35] M. Țălu, "Performance Evaluation of DNA-Based Cryptographic Algorithms on Constrained IoT Devices," JUTI: Jurnal Ilmiah Teknologi Informasi, vol. 24, pp. 132–148, 2026. DOI: 10.12962/j24068535.v24i1.a1386.
[36] M. Țălu, " Security in Internet of Things and Cloud Computing Convergence: Current Trends, Challenges, and Perspectives," ANNALS of Faculty of Engineering Hunedoara, vol. 23, pp. 39–52, 2025.