Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Lightweight Cryptographic Algorithms for Resource-Constrained IoT Devices: A Security and Performance Evaluation

Aditya Kumar
M.S Cybersecurity
Department of Information Technology
University of Mumbai
Mumbai, India

Jayesh Shinde
Professor
Department of Information Technology
University of Mumbai
Mumbai India

Abstract - The rapid expansion of Internet of Things (IoT) systems has introduced significant security challenges due to the constrained computational, memory, and energy resources of embedded devices. Conventional cryptographic algorithms are often unsuitable for such environments, necessitating the use of lightweight cryptographic primitives optimized for efficiency. This paper presents a systematic security and performance evaluation of selected lightweight cryptographic algorithms under an embedded execution environment. Specifically, SIMON-64/128, PRESENT-80, and ASCON-128 are implemented in C and evaluated on an ESP32 microcontroller based on the Xtensa LX6 architecture using the ESP-IDF framework.

Performance evaluation is conducted using cycle-accurate hardware measurements, including execution time, CPU cycles, cycles-per-byte (CPB), and throughput under different compiler optimization levels (-Og and -O2). In addition, security properties are analyzed using diffusion-based metrics such as avalanche effect and key sensitivity. Experimental results demonstrate that SIMON achieves the highest computational efficiency, while PRESENT exhibits strong diffusion at the cost of significantly higher execution time. ASCON shows moderate performance but reveals an unexpected performance regression under higher compiler optimization, highlighting the impact of architecture-compiler interactions. Furthermore, the study identifies practical deployment constraints, including Task Watchdog Timer limitations affecting PRESENT in real-time embedded environments.

The findings emphasize that algorithm suitability in IoT systems depends not only on theoretical design but also on implementation-level behavior under specific hardware and compiler configurations. This work provides a reproducible evaluation framework and offers practical insights for selecting and optimizing lightweight cryptographic algorithms in resource-constrained embedded systems, forming a foundation for future research toward the design of more efficient lightweight encryption schemes.

Keywords-Lightweight Cryptography, Internet of Things, Security Evaluation, Performance Analysis, Resource-Constrained Device

I. INTRODUCTION

A. Internet of Things and the Security Imperative

The Internet of Things (IoT) has become a core infrastructure layer across a wide range of application domains, including industrial automation, healthcare monitoring, smart grid management, vehicular communication networks, and environmental sensing systems. The number of globally connected IoT endpoints is projected to surpass 29 billion devices by 2030 [1], resulting in continuous and large-scale exchange of sensitive operational data across heterogeneous, often unattended networks. This scale of deployment introduces security requirements that are both technically demanding and operationally critical. Data confidentiality must be maintained across untrusted communication channels, message integrity must be guaranteed against active adversaries, and device authenticity must be verifiable without dependence on centralized infrastructure.

These requirements are well-addressed by conventional cryptographic standards in traditional computing environments. The Advanced Encryption Standard (AES) and RSA-based public-key schemes provide robust, formally analyzed security guarantees and are extensively deployed across enterprise and cloud platforms. However, the direct application of such algorithms to IoT endpoints is fundamentally constrained by the physical and architectural characteristics of these devices. Resource-constrained IoT devices possess limited computational, memory, and energy capabilities, making conventional cryptographic algorithms inefficient for real-time deployment [2], [3].

B. Lightweight Cryptography: Scope and Standardization

Lightweight cryptography has emerged as the primary response to the security limitations of conventional algorithms in constrained environments. Lightweight cryptographic primitives are designed to minimize computational complexity, reduce memory footprint, and limit energy consumption, while maintaining security levels appropriate for the threat models relevant to IoT deployments. The field covers a broad range of primitive types, including lightweight block ciphers, stream

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

ciphers, hash functions, and authenticated encryption with associated data (AEAD) schemes.
ASCON was standardized by NIST as a lightweight cryptographic standard for constrained environments [4]. This selection followed an exhaustive evaluation process covering security analysis, hardware efficiency, and software performance across a diverse range of target platforms. Alongside ASCON, algorithms such as SIMON and PRESENT have remained widely studied due to their architectural simplicity and demonstrated suitability for constrained software environments. SIMON, published by the National Security Agency (NSA) in June 2013 [5], employs a balanced Feistel structure using bitwise rotation, AND, and XOR operations, making it well suited to software implementation on general-purpose processors. PRESENT, originally proposed by Bogdanov et al. at the Workshop on Cryptographic Hardware and Embedded Systems in 2007 [6] and subsequently standardized under ISO/IEC 29192-2 [7], achieves strong diffusion through a bit-level permutation layer, an operation with significant implications for software execution efficiency on register-based processor architectures.

Despite the volume of existing theoretical and simulation-based evaluations, a critical gap remains in the literature. Systematic, cycle-accurate benchmarking of these algorithms under real embedded hardware conditions, with explicit consideration of compiler optimization behavior and operational firmware constraints, has not been adequately addressed in prior work.

C. Limitations of Existing Evaluation Approaches

A review of the existing lightweight cryptography evaluation literature reveals three distinct categories of limitation that collectively reduce the practical utility of published results for real-world IoT deployment decisions.

The first limitation concerns the evaluation environment. A substantial proportion of published performance evaluations are conducted using high-level programming languages such as Python or Java, executing on general-purpose desktop or server hardware [8]. While this approach offers reproducibility and ease of implementation, it does not accurately reflect the execution behavior of these algorithms on embedded target hardware. Interpreter overhead, dynamic memory allocation, garbage collection, and the absence of hardware-level cycle counting introduce measurement artifacts that distort the relative performance rankings of algorithms whose efficiency profiles are inherently architecture-dependent.
The second limitation concerns the choice of performance metrics. Existing studies that do employ embedded hardware typically report execution time as the sole performance indicator, measured through software timers of limited resolution. Cycle-accurate profiling, which isolates algorithmic complexity from clock frequency and enables meaningful comparison across processor architectures, is rarely reported. The omission of normalized metrics such as cycles-per-byte (CPB) further limits cross-study comparability, since raw execution times are inherently dependent on the operating frequency of the specific evaluation platform used.

The third limitation concerns compiler optimization. The impact of compiler optimization configuration on lightweight cipher performance has received minimal systematic attention in the existing literature. Compiler flags such as -Og and -O2 affect instruction selection, loop unrolling, register allocation, and function inlining in ways that interact non-trivially with the internal structure of cryptographic algorithms. For ciphers whose efficiency depends on bitwise operation throughput and register pressure, characteristics directly relevant to SIMON, PRESENT, and ASCON, the compiler optimization level constitutes a meaningful and independent experimental variable that must be explicitly controlled and reported.

D. This Study: Scope, Platform, and Contributions

This paper addresses the identified gaps through a bare-metal, cycle-accurate performance and security evaluation of three representative lightweight cryptographic algorithms, namely SIMON-64/128, PRESENT-80, and ASCON-128, implemented in C and executed on the ESP32 microcontroller platform running on an Xtensa LX6 processor at 160 MHz. The ESP32 was selected as the evaluation platform due to its widespread deployment in production IoT systems, its 32-bit Xtensa architecture representative of mid-range IoT processors, and the availability of ESP-IDF v6.0 as a controlled and reproducible firmware environment. All implementations were evaluated under two compiler optimization configurations, a debug build using -Og and a performance build using -O2, enabling systematic analysis of optimization sensitivity as an independent experimental variable.
This study presents a cycle-accurate bare-metal evaluation of SIMON-64/128, PRESENT-80, and ASCON-128 on the ESP32 Xtensa LX6 platform under multiple compiler optimization levels. The work analyzes execution time, CPU cycles, throughput, memory footprint, avalanche effect, and key sensitivity while identifying architecture-specific behaviors including ASCON optimization regression and operational constraints affecting PRESENT under FreeRTOS.

The remainder of this paper is organized as follows. Section II reviews existing literature on lightweight cryptographic performance and security evaluation. Section III describes the experimental methodology, hardware platform, and implementation details. Section IV presents and analyzes the performance and security results. Section V discusses the architectural implications and deployment recommendations derived from the findings. Section VI concludes the paper and outlines directions for future research.

II. LITERATURE REVIEW

A. Overview of Lightweight Cryptographic Algorithms

The rapid growth of IoT deployments has driven sustained research interest in lightweight cryptography as a distinct and specialized branch of applied cryptography. Unlike conventional cryptographic standards designed for high-performance computing platforms, lightweight cryptographic primitives are specifically engineered to operate within the strict computational, memory, and energy budgets of constrained devices such as RFID tags, wireless sensor nodes, and embedded microcontrollers [9], [10].

Several studies have surveyed lightweight cryptographic algorithms for IoT environments, highlighting trade-offs between computational efficiency, memory usage, and security robustness [3], [9], [11]. However, most existing works lack

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

cycle-accurate embedded benchmarking and compiler optimization analysis.

Existing research broadly categorizes lightweight cryptographic primitives into block ciphers, stream ciphers, hash functions, and authenticated encryption with associated data schemes. Among block ciphers, the Feistel-based SIMON family [5] and the Substitution-Permutation Network-based PRESENT cipher [6] have received sustained academic attention due to their architectural simplicity and demonstrated suitability for constrained environments. In the AEAD category, ASCON has emerged as the dominant standard following its selection by NIST in 2023 and its formal publication as NIST Special Publication 800-232 [4]. Khan et al. further contextualized lightweight cryptographic algorithms within a protocol-level framework, noting that the asymmetric computational nature of IoT systems requires elastic security solutions capable of adapting to varying resource availability across end, edge, fog, and cloud tiers [12].

B. Existing Performance and Security Evaluation Studies

A substantial body of research has focused on benchmarking lightweight cryptographic algorithms across different implementation environments. Goyal et al. evaluated PRESENT, AES, ECDH, DH, and RSA in hardware using a UMC-90 nm standard gate library, reporting a 1.7x improvement in area and a 63x improvement in power consumption for a modified PRESENT design compared to AES [13]. While this work established PRESENT as a viable hardware candidate for constrained devices, the study relied entirely on hardware implementation metrics and did not address software execution on general-purpose embedded processors, leaving the question of PRESENT's software efficiency unanswered.

El-Hajj and Fadlallah conducted one of the more extensive hardware-based benchmarking studies, evaluating 39 block ciphers on an ATMEGA328p Arduino microcontroller and Raspberry Pi, later extending the comparison to include 80 NIST second-round candidates [14]. Their results confirmed that performance rankings vary significantly across hardware platforms and that architecture directly influences algorithm ordering. However, both evaluation platforms used in that study - an 8-bit AVR microcontroller and a Linux-based single-board computer - differ substantially from the 32-bit Xtensa LX6 architecture that forms the core of widely deployed IoT systems such as the ESP32. This platform gap means their conclusions cannot be directly extrapolated to production IoT firmware environments.

Radhakrishnan et al. compared AES-128, SPECK, and ASCON on constrained IoT boards, measuring execution time, memory utilization, latency, throughput, and security robustness [8]. SPECK was found to exhibit the best overall performance on resource-constrained devices. While this study represents a meaningful step toward practical hardware evaluation, the implementation was conducted using high-level language tools rather than bare-metal C, which introduces interpreter overhead and limits cycle-accurate analysis. Compiler optimization configurations were not treated as an independent variable, and SIMON and PRESENT were not included in the comparison.

Sorescu et al. performed a comparative analysis on Nordic Semiconductor nRF5340 platforms using ARM Cortex-M33 processors, further confirming that performance rankings are platform-specific and that no single algorithm dominates across all tested metrics [15]. Similarly, a more recent study evaluated SPECK and SIMON on ESP32 and STM32 microcontrollers in the context of Internet of Drones security, finding that SPECK outperformed other evaluated ciphers on ESP32 while noting architectural differences between the two platforms [16]. Importantly, that study used an Arduino framework rather than bare-metal ESP-IDF, meaning that the underlying compiler optimization behavior and hardware resource usage were not directly observable. To the best of the authors' knowledge, no prior study has conducted a cycle-accurate bare-metal evaluation of SIMON, PRESENT, and ASCON simultaneously on the Xtensa LX6 architecture under controlled compiler optimization conditions.

Vennela et al. and Ibrahim and Agbinya each contributed to the discussion of algorithm suitability from a design and comparison perspective, with the latter proposing a novel SLA cipher based on an SPN structure and demonstrating higher throughput compared to existing ciphers in a Python simulation environment [10], [17]. While such design-oriented studies are valuable in expanding the solution space, their Python-based implementations introduce significant interpreter overhead that prevents meaningful comparison with bare-metal embedded results.

C. Survey Studies and Standardization Context

Several survey studies have established the foundational understanding of lightweight cryptography for IoT environments. Buchanan et al. and Thakor et al. [11], [3] provided early comprehensive analyses of lightweight cryptographic algorithms and their trade-offs in constrained systems. Ouaissa et al. further surveyed IoT edge security mechanisms involving ASCON, PRESENT, SIMON, and SPECK, emphasizing that conventional algorithms such as AES and RSA impose excessive computational and energy overhead for constrained devices [2].

Khan et al. highlighted that protocol-level overhead significantly influences the practical suitability of lightweight cryptographic schemes in IoT deployments [12]. Similarly, Qaid et al. identified efficiency, security, key generation, S-box design, and resource utilization as key evaluation parameters for battery-constrained IoT devices, concluding that no single algorithm simultaneously optimizes all criteria [18].

A PRISMA-based review by Khan et al. covering Industrial IoT systems further emphasized that real-time operational constraints are frequently overlooked in existing evaluations [19]. This observation directly relates to the PRESENT-80 Task Watchdog Timer behavior observed in this study under ESP32 FreeRTOS execution.

Additional research has explored application-specific lightweight cryptographic solutions, including medical IoT security mechanisms [20] and blockchain-integrated IoT frameworks using PRESENT and ECC [21]. SIMON, PRESENT, and ASCON continue to represent important Feistel,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

SPN, and permutation-based lightweight cryptographic design approaches respectively [4]–[7], with prior studies reporting strong architecture-dependent performance variation across platforms [15], [24].

D. Compiler Optimization and Embedded Implementation Studies

The role of compiler optimization in lightweight cryptographic performance has received limited but growing attention. Kim et al. investigated whether assembly or compiler-optimized C produces better performance for lightweight block ciphers on 32-bit RISC-V processors, demonstrating a 128.5% improvement in PIPO cipher performance using optimized C over a naive port [25]. Their work establishes that compiler optimization level is a meaningful and non-trivial experimental variable for embedded cryptographic implementations, supporting the rationale for treating -Og and -O2 as independent variables in this study.

Work on general embedded systems confirms that GCC optimization levels significantly affect execution time and energy on constrained platforms, with optimized programs outperforming unoptimized versions by factors ranging from 2.3x to 36.7x depending on the algorithm and target architecture [26]. Critically, this impact is architecture-dependent, meaning that optimization sensitivity observed on one processor family cannot be assumed to transfer to another. This principle directly motivates the Xtensa-specific optimization analysis in this paper and provides a theoretical foundation for interpreting the ASCON -O2 regression as an architecture-compiler interaction effect rather than a general property of ASCON.

Plauska et al. evaluated multiple programming languages on ESP32, demonstrating that C/C++ consistently outperforms interpreted languages such as MicroPython by substantial margins on the Xtensa LX6 platform [27]. This result reinforces the experimental design choice of bare-metal C implementation in this study and confirms that Python-based evaluations of cryptographic algorithms on ESP32-class hardware introduce significant measurement artifacts that distort performance rankings.

E. Limitations of Existing Research and Positioning of This Study

Three consistent limitations emerge from reviewing the existing literature. First, a substantial proportion of performance evaluations are conducted either in hardware simulation environments or using high-level language implementations on general-purpose computing hardware, neither of which accurately reflects the execution behavior of lightweight algorithms on production embedded firmware. Second, the Xtensa LX6 architecture, despite being one of the most widely deployed 32-bit IoT processors through the ESP32 platform, has received minimal attention as an evaluation target for lightweight cryptographic benchmarking. Third, compiler optimization configuration has not been systematically treated as an independent experimental variable in any prior study evaluating SIMON, PRESENT, and ASCON together.

This paper addresses all three gaps through a bare-metal, cycle-accurate implementation and evaluation of SIMON-64/128, PRESENT-80, and ASCON-128 on the ESP32 platform using the ESP-IDF v6.0 toolchain under two compiler optimization configurations. The findings, including the ASCON -O2 regression and the PRESENT-80 Task Watchdog Timer violation under FreeRTOS, represent contributions that are only observable through the methodology adopted in this study.

III. METHODOLOGY

A. Research Design and Evaluation Approach

This study adopts an experimental evaluation methodology based on bare-metal implementation and cycle-accurate hardware profiling. Rather than relying on software simulation environments or high-level language interpreters, all cryptographic operations were executed directly on embedded hardware using a controlled firmware environment. The evaluation focuses on three widely recognized lightweight cryptographic algorithms - SIMON-64/128, PRESENT-80, and ASCON-128 - selected to represent three distinct design philosophies within the lightweight cryptography landscape: Feistel-based block ciphers, SPN-based block ciphers, and permutation-based authenticated encryption schemes respectively. Each algorithm was evaluated under two compiler optimization configurations, treating the optimization level as an independent experimental variable. This design enables systematic analysis of how compiler decisions interact with algorithmic structure on a specific 32-bit embedded architecture.

The overall experimental architecture used for evaluation is illustrated in Fig. 1.

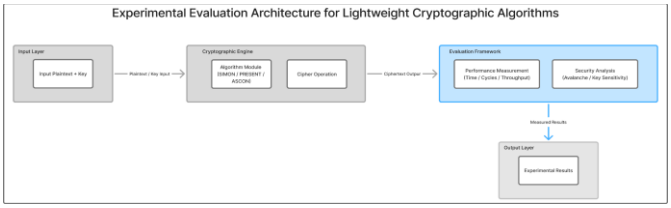

Fig. 1. Experimental evaluation architecture for lightweight cryptographic algorithms, illustrating the flow of data from plaintext and key input through cryptographic processing to performance and security evaluation, resulting in experimental results.

B. Hardware Platform

All experiments were conducted on an ESP32 microcontroller development board. The ESP32 integrates a dual-core Tensilica Xtensa LX6 processor and is one of the most widely deployed 32-bit microcontrollers in production IoT systems globally. For this evaluation, only a single core (CPU0) was used to ensure deterministic, non-preempted execution of benchmark workloads. The relevant hardware specifications are as follows.

The processor operates at a fixed clock frequency of 160 MHz, as confirmed by the boot log entry: *cpu freq: 160000000 Hz*. The chip revision is v3.0, identified from the efuse initialization log. Available memory resources include 520 KB of SRAM, of which 131,072 bytes (128 KB) are mapped as IRAM for instruction-critical code, and 180,736 bytes (approximately 176 KB) are available as DRAM. Flash storage uses DIO mode on a GD-series SPI flash chip. The platform does not include dedicated cryptographic hardware acceleration, meaning all

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

cipher operations are executed entirely in software on the general-purpose Xtensa LX6 processor. This is an important distinction, as any measured performance reflects pure algorithmic efficiency on the target architecture rather than hardware-assisted throughput.

The ESP32 was selected for several reasons beyond its widespread deployment. Its 32-bit word-level register architecture creates a meaningful and practically relevant test environment for evaluating the software efficiency of both word-friendly ciphers such as SIMON and bit-oriented ciphers such as PRESENT. The availability of ESP-IDF v6.0 as a mature, reproducible firmware framework further ensures that build configurations can be precisely controlled and documented.

C. *Software Environment and Firmware Configuration*

The firmware was developed using Espressif IoT Development Framework version 6.0 (ESP-IDF v6.0) with FreeRTOS as the underlying execution environment. Benchmarks were executed on a dedicated CPU0 task with background wireless and peripheral services disabled to minimize runtime interference.
All cryptographic operations and performance measurements were implemented in the C programming language. This choice was motivated by prior work demonstrating that interpreted environments such as MicroPython introduce significant runtime overhead on the Xtensa LX6 platform, reducing the reliability of cryptographic benchmarking results [29].
Two firmware builds were evaluated using the `-Og` and `-O2` compiler optimization levels. Each configuration was compiled and benchmarked independently to analyze optimization sensitivity across algorithms. The total binary image size decreased from 140,385 bytes under `-Og` to 134,091 bytes under `-O2`, indicating improved code generation efficiency under optimization

D. *Algorithm Implementations*

SIMON-64/128 was implemented from scratch in bare-metal C following the original NSA specification [5]. The implementation uses a 44-round Feistel structure based on rotation, AND, and XOR operations with an LFSR-based key schedule. Cipher outputs were stored in volatile memory to prevent compiler dead-code elimination, and output verification confirmed correct implementation behavior.
PRESENT-80 was implemented in bare-metal C using the standard 31-round SPN structure defined in the original PRESENT specification [6]. The implementation uses a software-based bit permutation layer operating across all 64 state bits, which contributes significantly to computational overhead on the Xtensa LX6 architecture due to the absence of dedicated permutation instructions. Output verification confirmed implementation correctness. Benchmark execution was limited to 100 iterations to avoid triggering the ESP32 Task Watchdog Timer under FreeRTOS.
ASCON-128 was implemented using the official `ascon-c` reference implementation published by the ASCON design team [28]. The `CRYPTO_AEAD_ENCRYPT` API was used for authenticated encryption of a 16-byte plaintext without associated data. Volatile output buffers and a short stabilization delay were used to ensure reliable timing measurements, and output verification confirmed correct encryption behavior.

E. *Performance Measurement Methodology*

Performance was measured using two complementary hardware facilities provided by the ESP-IDF framework. Execution time was measured using *esp_timer_get_time()*, which returns elapsed time in microseconds using a hardware timer peripheral independent of the CPU clock. CPU cycle counts were measured using *esp_cpu_get_cycle_count()*, which reads the Xtensa processor's internal cycle counter register directly. This provides cycle-accurate profiling that is decoupled from the system timer and unaffected by interrupt latency, making it the primary metric for cross-configuration comparison.

Both measurements were taken as a pair at the start and end of each benchmark loop. SIMON-64/128 and ASCON-128 were each executed for 1000 iterations, with the total elapsed time and total cycle count divided by the number of iterations to produce per-operation averages. PRESENT-80 was executed for 100 iterations due to the WDT constraint described above. Average execution time in microseconds and average CPU cycles per operation are reported as primary metrics. Throughput in kilobits per second is derived from the block size and average encryption time. Cycles-per-byte (CPB) is computed by dividing average CPU cycles by the block size in bytes, providing a normalized efficiency metric that is independent of clock frequency and enables fair cross-architecture comparison.

F. *Security Evaluation Methodology*

Security properties were evaluated using two diffusion-based metrics: avalanche effect and key sensitivity. Both metrics were computed using a Python-based evaluation environment operating on verified reference implementations of each algorithm. The avalanche effect measures the percentage of output bits that change when a single plaintext bit is flipped, averaged across all bit positions. An ideal block cipher approaches 50%, indicating strong diffusion. Key sensitivity measures the average percentage of output bits that change when a single key bit is flipped while the plaintext remains constant. These metrics provide a fundamental but widely accepted indication of resistance to differential analysis.

It should be noted that while the hardware performance measurements were obtained on the ESP32 target, the security diffusion metrics were computed separately in a controlled software environment. Cryptographic diffusion properties are mathematical characteristics of the algorithm itself and are not affected by the execution platform. This separation is methodologically valid and is consistent with standard practice in the lightweight cryptography evaluation literature [8].

The experimental workflow followed during benchmarking is illustrated in Fig. 2.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

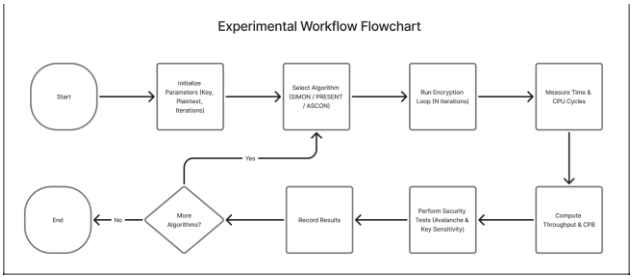

Fig. 2. Experimental workflow for iterative evaluation of lightweight cryptographic algorithms, including initialization, performance measurement, and security analysis.

G. *Testing Procedure Summary*

A consistent procedure was followed across all evaluated algorithms to ensure comparability. Fixed plaintext and key values were used throughout to eliminate variability due to input randomness. High-resolution hardware timing and cycle counting were performed in the same benchmark loop to minimize measurement overhead. Results were recorded from the serial console output and manually transcribed. All experiments were conducted on a single device with no concurrent wireless or Bluetooth activity. The complete source code for all implementations, build configurations, and benchmark scripts used in this study is publicly available at: https://github.com/UNRIVALLEDKING/Lightweight-Crypto-ESP32-Benchmark [29].

IV. Results and Discussion

This section presents and discusses the experimental results obtained from the performance and security evaluation of the selected lightweight cryptographic algorithms. The analysis is based on implementation-level measurements.

A. *Performance Analysis*

Table I presents the performance results obtained from the ESP32 bare-metal evaluation under both compiler optimization configurations. All timing values are averaged over the full iteration count for each algorithm.

Table I. Performance Evaluation Results - ESP32 Xtensa LX6 at 160 MHz

Algorithm	Build	Avg Time (µs)	Avg Cycles	CPB	Throughput (Kbps)
SIMON-64/128	-Og	89.50	14,319	1,790	715.1
SIMON-64/128	-O2	82.91	13,265	1,658	771.9
PRESENT-80	-Og	11,462.03	1,833,923	229,240	5.58
PRESENT-80	-O2	9,829.19	1,572,664	196,583	6.51
ASCON-128	-Og	565.10	90,416	5,651	226.5
ASCON-128	-O2	613.82	98,211	6,138	208.5

CPB = Cycles Per Byte. Block sizes: SIMON 8 bytes, PRESENT 8 bytes, ASCON 16 bytes.

To clearly illustrate the relative computational cost across algorithms, a normalized execution time comparison is presented.

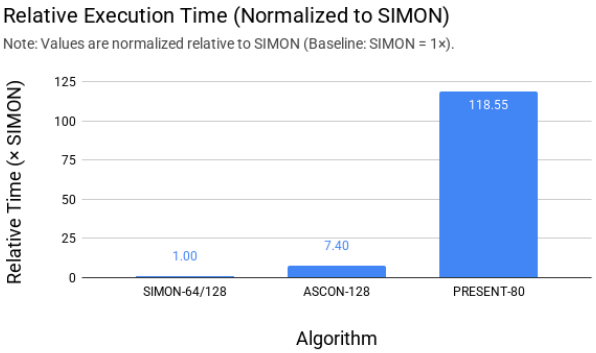

Fig. 3. Relative execution time of lightweight cryptographic algorithms normalized to SIMON (baseline = 1×).

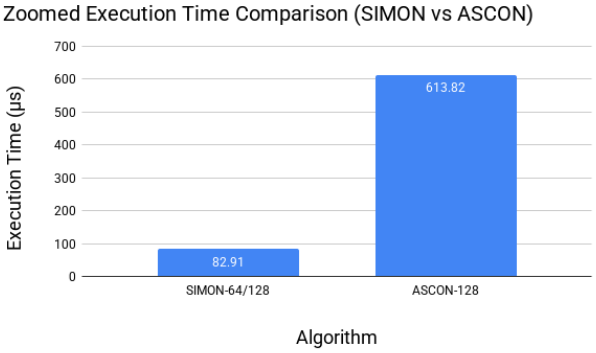

Fig. 4. Zoomed execution time comparison between SIMON and ASCON, illustrating finer-grained performance differences without the influence of PRESENT's significantly higher execution time.

Table II. Memory Footprint - ESP-IDF Build Analysis (-O2 Build)

Memory Region	Used (bytes)	Used (%)	Remaining (bytes)	Total (bytes)
Flash Code (.text)	48,788	-	-	-
IRAM (.text + .vectors)	40,759	31.1%	90,313	131,072

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Flash Data (.rodata + .appdesc)	34,112	-	-	-
DRAM (.data + .bss)	12,848	7.11%	167,888	180,736
RTC SLOW	64	0.78%	8,128	8,192
Total Image Size	134,091			

-Og build total image size: 140,385 bytes. Savings under -O2: 6,294 bytes.

SIMON-64/128 Performance

SIMON-64/128 achieved the lowest execution time among all evaluated algorithms under both compiler configurations. Under the debug build, the average encryption time was 89.50 µs consuming 14,319 CPU cycles, equivalent to a throughput of 715.1 Kbps and a CPB of 1,790. Under the performance build, these figures improved to 82.91 µs, 13,265 cycles, 771.9 Kbps throughput, and a CPB of 1,658, representing a 7.3% reduction in execution time. This consistent and predictable improvement under -O2 is directly attributable to SIMON's use of simple bitwise rotation, AND, and XOR operations, which map efficiently to the Xtensa LX6 instruction set and benefit straightforwardly from compiler instruction scheduling and register allocation optimizations. The output verification value of 0xa9291196 confirmed correct cipher operation across all 1000 iterations.

These results confirm that SIMON-64/128 is the most computationally efficient algorithm among those evaluated for software-based IoT deployments on 32-bit Xtensa processors, consistent with findings from prior studies on other architectures [15], [26].

The CPU cycle consumption across algorithms further highlights this difference, as shown in Fig. 5.

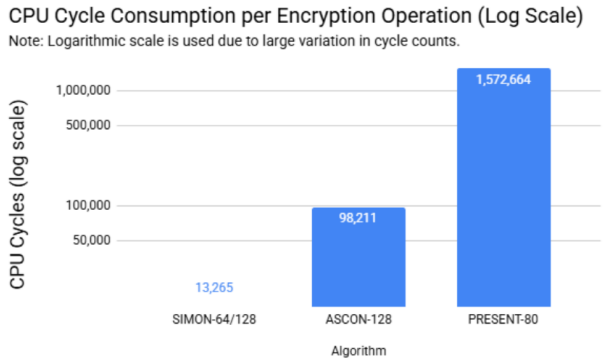

Fig. 5. CPU cycle consumption per encryption operation for lightweight cryptographic algorithms. A logarithmic scale is used to accommodate large variations in cycle counts across implementations.

PRESENT-80 Performance

PRESENT-80 produced execution times several orders of magnitude higher than SIMON under both builds. The debug build averaged 11,462.03 µs per encryption operation at 1,833,923 CPU cycles, giving a CPB of 229,240 and a throughput of just 5.58 Kbps. Under the performance build, this improved to 9,829.19 µs and 1,572,664 cycles - a 14.2% reduction. While this represents the largest absolute improvement in raw execution time among the three algorithms, PRESENT-80 remains by far the least efficient algorithm in this study.

The fundamental cause of this overhead is the bit-level permutation layer in PRESENT's SPN structure. Each of the 31 rounds requires individually extracting and repositioning all 64 bits of the cipher state through a software loop, an operation that requires 64 sequential bit manipulations per round with no possibility of word-level parallelism on a 32-bit processor without bit-slicing techniques. Kim et al. demonstrated the same phenomenon with the SPEEDY cipher, showing that hardware-oriented bit-permutation ciphers consistently achieve poor CPB performance on 32-bit MCUs in software without architectural optimizations [24]. PRESENT's design was explicitly optimized for hardware gate efficiency, and this study confirms that this optimization comes at a significant cost in software environments.

More critically, PRESENT-80 triggered the ESP32 Task Watchdog Timer (WDT) under the default FreeRTOS configuration in both compiler build variants. With average execution time approaching 9.8 ms per encryption operation under -O2, even moderate benchmark iteration counts approach watchdog timeout thresholds. This behavior introduces substantial deployment challenges for latency-sensitive real-time cryptographic workloads on standard FreeRTOS-based IoT firmware unless additional mitigation strategies—such as hardware acceleration, periodic WDT feed operations, or modified watchdog configurations—are employed. This operational constraint is not typically observable in simulation-based or high-level language evaluations and highlights the importance of bare-metal embedded testing for accurately assessing real-world lightweight cryptographic deployment behavior.

ASCON-128 Performance

ASCON-128 achieved moderate performance between SIMON and PRESENT. Under the debug build, average execution time was 565.10 µs at 90,416 cycles, giving a CPB of 5,651 and a throughput of 226.5 Kbps. The output verification byte of 0x08 confirmed correct authenticated encryption across all 1000 iterations.

The most notable observation in the ASCON results is the performance regression under -O2 optimization. Rather than improving as expected, ASCON-128 became slower under the performance build at 613.82 µs and 98,211 cycles - an increase of 8.6% in execution time and 8.6% in cycle count compared to the debug build. This behavior, referred to in this paper as the ASCON -O2 Regression, is contrary to the expected direction of compiler optimization and requires careful interpretation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

The likely explanation lies in the interaction between ASCON's internal permutation structure and the Xtensa LX6 compiler backend under -O2. ASCON's permutation operates on a 320-bit state organized as five 64-bit words, applying a sequence of XOR, AND, and rotation operations in multiple rounds. Under -O2, the compiler's inlining and loop unrolling decisions may increase IRAM pressure by expanding the code footprint of hot inner loops beyond what fits in the Xtensa instruction cache efficiently, causing cache pressure effects that outweigh the benefits of reduced loop overhead. An alternative contributing factor is that -O2 register allocation decisions for the permutation's 64-bit word operations may increase spill-and-reload cycles relative to the more conservative -Og scheduling. Distinguishing between these causes would require architecture-level profiling beyond the scope of this study, but the finding itself is documented and reproducible from the published benchmark repository [30].

This result has a direct practical implication: engineers deploying ASCON on Xtensa-based IoT systems should not assume that selecting a higher compiler optimization level will improve performance. Empirical validation under the target build configuration is necessary.

A comparative view of throughput across all evaluated algorithms is shown in Fig. 6.

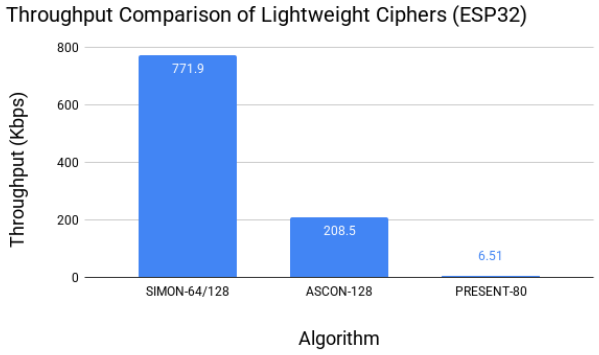

Fig. 6. Throughput comparison of lightweight cryptographic algorithms on the ESP32 platform, showing SIMON achieving the highest efficiency, followed by ASCON, while PRESENT exhibits significantly lower throughput.

B. Security Analysis

Table III presents the security diffusion results for all evaluated algorithms.

Table III. Security Evaluation Results

Algorithm	Avalanche Effect (%)	Key Sensitivity (%)
SIMON-64/128	50.98	49.77
PRESENT-80	50.44	49.92
PRESENT-128	51.12	50.23
ASCON-128	1.56	49.40

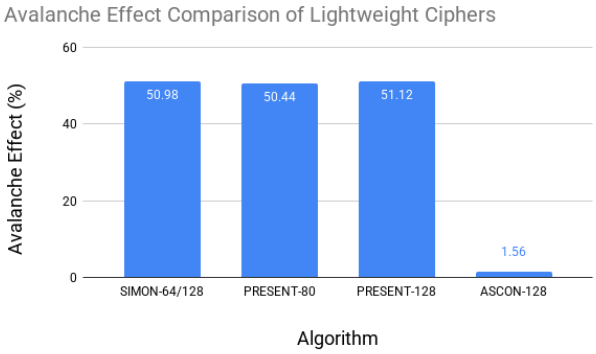

Fig. 7. Avalanche effect comparison of lightweight cryptographic algorithms. SIMON and PRESENT exhibit strong diffusion properties close to the ideal 50%, while ASCON shows significantly lower diffusion under single-block evaluation due to its permutation-based design.

The diffusion characteristics of the evaluated algorithms are further illustrated in Fig. 7.
SIMON-64/128 and both PRESENT variants achieve avalanche effect values close to the theoretical ideal of 50%, demonstrating strong diffusion properties and confirming that single-bit changes in the plaintext propagate effectively across the full ciphertext output. Key sensitivity values similarly approach 50% for all block cipher designs, indicating robust dependence on key variation and resistance to related-key effects.

ASCON-128 produces a markedly low avalanche effect of 1.56% under the single-block evaluation model used in this study. This result requires careful interpretation and does not indicate cryptographic weakness. ASCON is a permutation-based AEAD scheme in which diffusion is distributed across an internal 320-bit sponge state and accumulates meaningfully over multiple message blocks and initialization rounds. The single-block avalanche test captures only a partial view of ASCON's diffusion behavior and is not a metric for which ASCON was designed to optimize. This limitation in applying block-cipher-oriented diffusion metrics to permutation-based designs is documented in the existing literature [8] and is discussed further in this paper's contribution to evaluation methodology. ASCON's key sensitivity value of 49.40% confirms that key-dependent output variation remains strong even in short-message scenarios.

C. Comparative Discussion

The results establish a clear three-tier performance hierarchy on the Xtensa LX6 architecture. SIMON-64/128 occupies the top tier with sub-100 µs encryption times and a CPB under 1,700 under -O2. ASCON-128 occupies the middle tier at approximately 600 µs and CPB around 6,100. PRESENT-80 occupies a distant bottom tier at nearly 10,000 µs and CPB exceeds 196,000 under -O2, separated from ASCON by a factor of roughly 16 in execution time.
The results further demonstrate that compiler optimization sensitivity is algorithm-specific and non-uniform. SIMON exhibits a predictable 7.3% performance improvement under –

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

`-O2`, consistent with its software-oriented design characteristics. PRESENT achieves a 14.2% reduction in execution time under optimization; however, it continues to exhibit significant practical limitations for real-time deployment on the evaluated ESP32 FreeRTOS configuration. In contrast, ASCON experiences an 8.6% performance regression under `-O2`, indicating an architecture-compiler interaction specific to the Xtensa LX6 platform and the `ascon-c` reference implementation. These observations demonstrate that compiler optimization behavior cannot be treated as a fixed or universally beneficial parameter in IoT cryptographic deployments. Instead, optimization effects must be empirically evaluated for each algorithm-platform-toolchain combination.

From a memory perspective, the -O2 build reduces total image size by 6,294 bytes compared to -Og. At 134,091 bytes total, the complete benchmark firmware including all three cipher implementations, the FreeRTOS kernel, and ESP-IDF system libraries consumes 31.1% of available IRAM and 7.11% of DRAM. This confirms that all three algorithms are memory-feasible on the ESP32 platform at the firmware level, and that the primary differentiation factor is computational efficiency rather than memory footprint.

V. CONCLUSION AND FUTURE SCOPE

A. *Summary of Findings*

This study presented a systematic security and performance evaluation of selected lightweight cryptographic algorithms-SIMON, PRESENT (80-bit and 128-bit variants), and ASCON-under a unified software-based experimental environment. The evaluation focused on implementation-level measurements using consistent performance and security metrics, including encryption time, decryption time, throughput, avalanche effect, and key sensitivity.

Experimental results demonstrated that SIMON achieves superior computational efficiency, exhibiting the lowest execution time and highest throughput among the evaluated algorithms. PRESENT, while comparatively slower, showed strong diffusion properties and consistent key sensitivity, particularly in its 128-bit variant. ASCON displayed moderate performance characteristics; however, its diffusion behavior under single-block avalanche testing differed significantly due to its permutation-based design. These findings highlight the inherent trade-offs between efficiency and diffusion strength across different lightweight cryptographic designs.

B. *Implications of Resource-Constrained IoT Devices*

The results of this evaluation provide practical insights into the suitability of lightweight cryptographic algorithms for resource-constrained IoT environments.

Algorithms such as SIMON are well suited for latency-sensitive and software-based deployments where computational efficiency is critical. PRESENT offers stronger diffusion characteristics, making it appropriate for scenarios where security robustness is prioritized over raw performance.

The evaluation also emphasizes that permutation-based algorithms such as ASCON should be assessed using context-appropriate metrics, particularly for applications involving authenticated encryption and multi-block data streams. These observations reinforce the need for careful algorithm selection based on deployment requirements, threat models, and device capabilities in IoT systems.

C. *Limitations of the Study*

While the experimental evaluation provides valuable comparative insights, certain limitations must be acknowledged. First, all experiments were conducted in an embedded execution environment using the ESP32 microcontroller platform, which may not fully represent the behavior of algorithms on highly constrained embedded hardware. Second, energy consumption was not directly measured and was inferred only through execution-time-related metrics. Third, the security evaluation was limited to diffusion-based indicators such as avalanche effect and key sensitivity, without formal cryptanalytic analysis.

Additionally, the application of block-cipher-oriented security metrics to permutation-based algorithms such as ASCON introduces inherent evaluation constraints, particularly for short-message scenarios.

D. *Future Research Directions*

Future work will focus on addressing the identified limitations and extending the findings of this study. A key research direction involves the design and proposal of a new lightweight cryptographic algorithm that balances computational efficiency and diffusion strength, specifically optimized for software-based IoT deployments. Further evaluation on constrained hardware platforms, such as microcontrollers and sensor nodes, will provide deeper insights into real-world performance and energy consumption.

Additional research may also explore adaptive or hybrid lightweight cryptographic designs and develop evaluation methodologies better suited for permutation-based and authenticated encryption schemes. The findings of this evaluation study serve as a foundation for such future research efforts and motivate the development of improved lightweight cryptographic solutions for emerging IoT applications.

REFERENCES

[1] M. Hatton, "Global IoT Forecast Report, 2021–2030," Transforma Insights, Jul. 2022.
[2] M. Ouaissa et al., "Securing IoT edge: a survey on lightweight cryptography, anonymous routing and communication protocol enhancements," Int. J. Inf. Secur., Springer, 2025.
[3] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," IEEE Access, vol. 8, pp. 28177–28193, 2020.
[4] NIST, "Ascon-Based Lightweight Cryptography Standards for Constrained Devices," NIST SP 800-232, Aug. 2025.
[5] R. Beaulieu et al., "The SIMON and SPECK families of lightweight block ciphers," Cryptology ePrint Archive, 2013.
[6] A. Bogdanov et al., "PRESENT: An ultra-lightweight block cipher," CHES 2007, pp. 450–466.
[7] ISO/IEC 29192-2, "Lightweight Cryptography - Block Ciphers," 2019.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

[8] I. Radhakrishnan et al., "Efficiency and security evaluation of lightweight cryptographic algorithms," Sensors, 2024.
[9] W. J. Buchanan et al., "Lightweight cryptography methods," J. Cyber Security Tech., 2017.
[10] V. Vennela et al., "Lightweight cryptography algorithms for IoT devices," IJRASET, 2021.
[11] P. S. Suryateja and K. Venkata Rao, "A survey on lightweight cryptographic algorithms in IoT," Cybernetics and Information Technologies, vol. 24, no. 1, pp. 1–15, 2024.
[12] M. N. Khan et al., "Lightweight cryptographic protocols for IoT-constrained devices," IEEE IoT J., 2021.
[13] T. K. Goyal et al., "Energy efficient lightweight cryptography algorithms," IETE J. Res., 2022.
[14] M. El-Hajj and A. Fadlallah, "Analysis of lightweight cryptographic algorithms on IoT hardware platforms," ITNAC, 2022.
[15] T. Sorescu et al., "Comparative performance analysis of lightweight cryptographic algorithms," 2025.
[16] "Design and assessment of lightweight cryptographic algorithms on ESP32," ScienceDirect, 2025.
[17] N. Ibrahim and J. Agbinya, "Lightweight cryptographic scheme for IoT devices," Sensors, 2023.
[18] G. R. S. Qaid et al., "Analysis of lightweight cryptographic algorithms for IoT," IJDSN, 2025.
[19] S. Khan et al., "Lightweight cryptographic mechanisms for IIoT systems," ACM Computing Surveys, 2025.
[20] A. M. Rasheed and R. M. S. Kumar, "Lightweight cryptography for medical IoT," IJACSA, 2024.
[21] S. Kamath et al., "Blockchain-based lightweight cryptographic scheme for IoT," IJAM, 2025.
[22] H. Kim et al., "Efficient implementation of SPEEDY cipher," Mathematics, 2022.
[23] M. Mirigaldi et al., "Efficient ASCON implementations," Chips, 2025.
[24] A. Sevin and A. Mohammed, "Survey on software implementation of lightweight block ciphers," JAIHC, 2023.
[25] H. Kim et al., "Assembly optimization for lightweight cryptography," ICISC, 2021.
[26] "Bit-level compiler optimization for ultra low-power systems," JSA, 2025.
[27] I. Plauska et al., "Programming language performance on ESP32," Electronics, 2023.
[28] ASCON Team, "ASCON-C Reference Implementation," GitHub, [Online]. Available: https://github.com/ascon/ascon-c
[29] A. Kumar, "Lightweight-Crypto-ESP32-Benchmark," GitHub Repository, 2026. Available:https://github.com/UNRIVALLEDKING/Lightweight-Crypto-ESP32-Benchmark