Performance Evaluation of Lightweight Cryptographic Ciphers on ARM Processor for IoT Deployments

Mohsin Khan(✉), Dag Johansen, and Håvard Dagenborg

UiT The Arctic University of Norway, Tromsø, Norway
{mohsin.khan,dag.johansen,havard.dagenborg}@uit.no

Abstract. The security mechanisms used to protect Internet-of-things and embedded systems often depend on cryptographic tools designed specifically to operate in resource-constrained environments. Several lightweight cryptographic ciphers have been proposed for such purposes, providing various properties and tradeoffs that balance security and performance. This paper provides a comprehensive evaluation of key lightweight ciphers, evaluating their throughput, processing efficiency, memory footprint, ROM utilization, and energy consumption. We focus on the software implementation and performance of these ciphers on a representative single-core ARM processor and propose an *E-Rank* metric that combines essential performance characteristics with resource consumption into a single comparative metric. The metric aids in identifying lightweight ciphers that achieve an optimal balance of efficiency by normalizing the trade-offs between performance, memory usage, and energy consumption.

Keywords: Lightweight cryptography · block cipher · stream cipher · performance evaluation · internet-of-things

1 Introduction

The widespread use of Internet of Thing (IoT) devices across various technological landscapes has emphasized the need for effective and sufficient security measures. One problem domain is securing the IoT devices themselves, where we previously have investigated secure enclave technologies supporting confidential computing [8,20]. Next, we have focused on securing the external IoT communication channels, where we conjecture that conventional cryptographic algorithms, such as, for instance, AES and RSA, are secure but less suitable for IoT applications due to their high computational and resource demands.

IoT devices generally operate with limited processing power, energy consumption, and memory, making it a challenge to balance security and efficiency. To address security challenges in IoT systems, Lightweight Cryptography (LWC) has become essential for providing integrity and confidentiality.

J. Zhao and W. Meng (Eds.): SciSec 2024, LNCS 15441, pp. 254–272, 2025.
https://doi.org/10.1007/978-981-96-2417-1_14

Lightweight Cryptographic Ciphers (LWCCs) are carefully designed to meet the unique demands and constraints of IoT environments, providing effective and adequate security solutions that protect devices without overburdening their limited computational resources or disrupting their operational efficiency. This strategic adaptation ensures that security implementations are both practical and effective for IoT applications.

This study offers an in-depth assessment of LWCCs across two primary categories: Lightweight Block Cipher (LWBC) and Lightweight Stream Cipher (LWSC) [13]. LWBCs, such as PRESENT [10], XTEA [19], CLEFIA [24], SIMON, and SPECK [5], are known for their ability to encrypt data blocks in constrained environments. On the other hand, LWSCs, including Mickey [2], Salsa [7], Sosemanuk [6], Grain-v1 [16], and an optimized version of Grain-128a [18], are designed for efficient encryption of continuous data streams.

The experiment is conducted on a Raspberry Pi Zero W, characterized by its single-core ARM11 processor. This preference is motivated by the device's widespread use in IoT applications due to its compact size, low power consumption, and ability to integrate with various sensors and actuators, thereby presenting a typical resource-constrained IoT environment. To measure the performance of each selected LWCC precisely, the study employs a custom benchmarking framework that evaluates throughput, Cycles per Byte (CpB), memory footprint, ROM utilization, and energy consumption. An Arduino UNO, coupled with a power measurement sensor, is used to measure the energy consumption during cipher operations. This provides insights into the energy consumption and efficiency of each lightweight cryptographic solution. Moreover, the study employs the *Rank* metric [5] and develops *E-Rank* that aggregates performance, resource usage, and energy consumption metrics into a single comparative value. This metric allows for a thorough evaluation of ciphers, aiding informed decision-making based on overall software performance.

The subsequent sections of this paper are structured as follows. The section on background will provide a concise overview of the selected LWCCs. The next section on research methodology explains our approach, including tool development, cipher selection criteria, hardware considerations, and evaluation metrics. The results and discussion section presents our findings and explores their implications. The related work section explains existing benchmarking tools briefly, highlighting their limitations and inapplicability to our research. Finally, the conclusions section summarizes key insights and suggests avenues for future research.

2 Background

2.1 Lightweight Block Ciphers

Significant progress has been made on LWBCs to accommodate the constraints of resource-limited environments. The Tiny Encryption Algorithm (TEA) [26] is characterized by its simplicity and the use of minimal cryptographic operations such as XOR, ADD, and SHIFT. However, vulnerabilities identified in TEA related to key-specific attacks [23] led to the development of an improved

version, XTEA [19], which improves security through dynamic key arrangement and delayed diffusion mechanisms. The National Security Agency (NSA) has contributed to this field as well by developing the SIMON and SPECK families of ciphers [4]. SIMON utilizes simple bitwise operations like AND, XOR, and circular shifts to achieve secure encryption in hardware environments. On the other hand, SPECK is designed to balance security and performance in software applications by using operations like modulo addition, XOR, and left and right circular shifts.

In 2007, the PRESENT [10] and CLEFIA [24] LWBCs were introduced and later standardized under ISO/IEC 29192 [17]. The PRESENT cipher has a compact design, achieved through the use of scan flip-flops and a single 4×4-bit S-Box with a linear mixing layer for permutation. Sony developed CLEFIA using a Diffusion Switching Mechanism (DSM), resulting in an increased number of active S-Boxes. The DSM operates by rotating the diffusion layers during encryption, increasing the number of active S-Boxes, and altering the interconnection patterns.

2.2 Lightweight Stream Ciphers

A significant initiative for LWSCs was the eSTREAM project, launched in 2004 under the European Network of Excellence for Cryptology (ECRYPT) [22]. Initially, 34 ciphers were presented, but only 7 were selected as the finalists. The seven ciphers were evaluated thoroughly in three phases and divided into two profiles: Profile 1 focused on software-oriented applications requiring high throughput, while Profile 2 focused on hardware-oriented applications with limited resources.

Profile 1 includes four cryptographic algorithms namely Rabbit [9], Salsa [7], HC-128 [27], and Sosemanuk [6]. Salsa operates by utilizing a pseudorandom function that is based on a hash function core. It uses a set of operations, including addition modulo, bitwise XOR, and constant distance rotation of 32 bits. Salsa20 is structured around a 20-round process that uses an invertible quarter-round function as its primary building block. Sosemanuk combines the elements of both synchronous stream ciphers and block ciphers, utilizing a Linear Feedback Shift Register (LFSR) and a Finite State Machine (FSM) to generate a keystream. The LFSR in Sosemanuk is responsible for providing a fast and efficient mechanism for generating bits, while the FSM handles the non-linear transformation of the state.

Profile 2 consists of Trivium [12], Grain [16], and Mickey [2]. Trivium uses three shift registers and a small amount of combinatorial logic. The cipher operates in two phases. During the initialization phase, the cipher loads the key and initialization vector (IV) into a 288-bit internal state, which then evolves over 1152 cycles. In the keystream generation phase, output bits are produced using XOR and AND operations on specific bits from the internal registers. Grain-v1 also features a two-phase operation. First, the Non-linear Feedback Shift Register (NLFSR) is loaded with a key, and the LFSR is initialized with an IV. In the second phase, feedback from both NLFSR and LFSR is used to generate

output bits through XOR operation with the input. Grain-128a is an improvement over Grain-v1, as it increases the key size to 128 bits and the internal state to 256 bits, making it more secure against vulnerabilities. Finally, the Mickey cipher improves randomness by clocking shift registers irregularly. Two registers are used, each comprising 80 stages, to create a keystream generator, with each stage controlling one bit.

3 Research Methodology

This study presents a structured approach for evaluating the software performance of LWCCs on Raspberry Pi Pico's ARM11 processor. Our methodology relies on a custom benchmarking tool to select, execute, and precisely measure cryptographic encryption and decryption operations for selected LWCCs. This tool evaluates the performance of ciphers using specific key metrics, including throughput, CpB, memory footprint, flash memory, and energy consumption. These metrics provide essential insights into the processor speed, memory efficiency, and energy utilization of the LWCCs, which are vital for their effective deployment in resource-constrained environments. During the selection process of the eleven ciphers, we carefully considered their relevance to IoT security challenges and their prominence within the lightweight cryptographic community.

3.1 Development of Benchmarking Tool

The benchmarking tool was developed using a systematic approach to ensure that it is lightweight, adaptable, and user-friendly. The tool was programmed using Python, which was extended with the C implementation of the ciphers through shared object files. This combination offers simple, flexible, and fast performance for both cryptographic operations and system monitoring. The following sections elaborate on the development process, providing a detailed account of each stage and methodology employed.

Requirement Analysis: The initial phase involved a thorough analysis of the requirements and objectives of the benchmarking tool. This included defining the desired features, input parameters, and output metrics. Based on the requirement analysis, a comprehensive design was developed for the benchmarking tool. The overview of the design of the tool and its associated workflow is illustrated in Fig. 1. The design of the tool involves defining its structure, including modules for input processing, cryptographic operations, and metrics related to performance measurement.

Implementation: The implementation phase is focused on translating the design and architecture into functional code. The core structure of the tool integrates the C implementations of the selected LWCCs with Python using the *ctypes* library [21], which enables the calling of C-implemented shared object

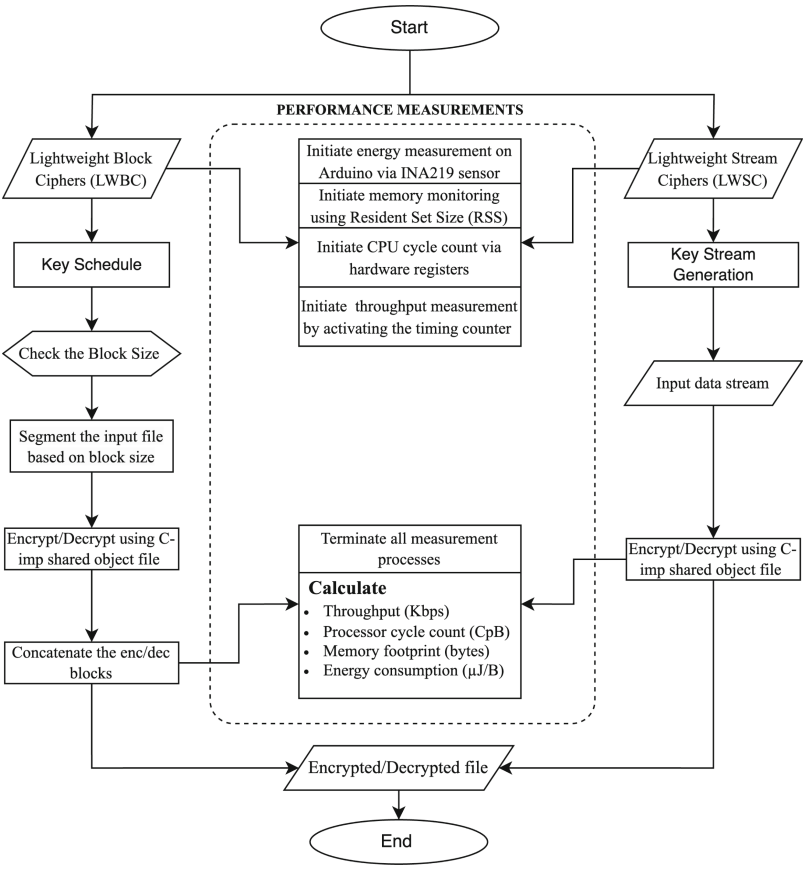


Fig. 1. Flowchart demonstrating basic encryption/decryption steps and associated initialization of performance tracking.

functions directly from Python. This hybrid approach leverages the speed and efficiency of C language for cryptographic operations while utilizing Python's robust capabilities for orchestrating the system procedures and data handling necessary to capture key performance metrics. Python was chosen for taking performance measurements due to its simplicity, extensive library ecosystem, and proven track record in rapid prototyping, significantly accelerating the development process. During this phase, several key functionalities were implemented:

- **Input Processing:** The program is designed to receive user inputs such as the block size, key size, and input file for encryption and decryption.
- **Cryptographic Operations:** Encryption and decryption are performed using specified parameters through the integration of a C implementation of selected LWCCs.
- **Performance Measurement:** The benchmarking tool incorporates libraries, functions, and system calls to measure key performance metrics like

throughput, CpB, memory usage, and energy consumption. This tool enables a comprehensive comparison of selected ciphers by evaluating the quantitative performance of the ciphers under standard conditions.
- **Result Analysis:** The data gathered from the benchmarking tool is used to generate bar graphs that demonstrate the performance of each LWCC across multiple metrics. To ensure the accuracy of these visual representations, the confidence interval is calculated at a 95% confidence level by measuring the performance of each cipher repeatedly across several iterations. This statistical approach provides a better understanding of the performance consistency and potential variability, allowing for a more fine interpretation of the data. By including these confidence intervals in the bar graphs, the analysis not only highlights average performance metrics but also provides insights into the stability and predictability of each cipher.

Testing and Debugging: During this phase, a set of tests were conducted. The tests include regression testing to ensure that any new changes made did not interfere with the existing functionality. Statistical testing was also conducted to verify the accuracy of data handling and analysis. Lastly, load testing was carried out to evaluate how the system would perform under various stress conditions. The tool was thoroughly validated against established benchmarks and reference implementations to ensure its accuracy and effectiveness. Feedback collected during this phase was vital in identifying and resolving any problems, bugs, or discrepancies, resulting in improved overall functionality and performance of the tool.

Optimization and Performance Tuning: Optimization techniques were applied to enhance both the efficiency and performance of the benchmarking tool. This process involved using only efficient, valid C implementations of LWCCs to execute core cryptographic operations with maximum speed and low-level access. Furthermore, to enhance the performance measurement and minimize resource usage, we have taken a careful approach to selecting and utilizing essential libraries, thereby reducing unnecessary overhead. These optimizations improved the tool's accuracy and responsiveness while keeping its footprint lightweight. This is particularly important when evaluating performance in environments with limited resources, ensuring reliable data delivery.

3.2 Selection Criteria for Lightweight Cryptographic Ciphers

In this study, eleven LWCCs were selected for evaluation, primarily due to their relevance to embedded system security, their prominence within the research community, and their availability in open-source cryptographic libraries. The ciphers selected include PRESENT, XTEA, CLEFIA, SIMON, and SPECK among LWBC, and Grain-v1, Grain-128a, Trivium, Mickey, Salsa, and Sosemanuk among LWSC.

PRESENT and CLEFIA were chosen based on their recognition in the lightweight cryptography community and their compliance with the ISO/IEC 29192 standard [17]. XTEA was included due to its simple and compact structure and its development at the Cambridge Computer Laboratory, marking it as one of the pioneering ciphers with a small footprint suitable for constrained environments. SIMON and SPECK were selected because of their development by the researchers at the National Security Agency (NSA).

Grain-v1, Trivium, Mickey, Salsa, and Sosemanuk were all finalists in the ECRYPT Stream Cipher Project, highlighting their cryptographic robustness and efficiency. We also chose a speed-optimized version of Grain-128a, which includes enhanced features such as added authentication, to evaluate potential performance improvements.

This diverse selection of LWCCs allows for a comprehensive analysis of cryptographic techniques applicable to IoT devices in terms of their computational efficiency and performance evaluation.

3.3 Implementation Considerations for ARM-Processor

The benchmarking tool has been specifically optimized for deployment on the Raspberry Pi Zero W. This device is equipped with a single-core ARM1176JZF-S processor [1]. The processor was chosen due to its specific computational capabilities and inherent memory constraints. These constraints are important in simulating the performance of LWCCs in constrained environments.

The Raspberry Pi Zero W features 512 MB of RAM, which provides efficient memory management to prevent bottlenecks during the execution of selected ciphers. The 40-pin header on the Raspberry Pi Zero W is utilized strategically, with designated pins specifically used for supplying power to the device and for data transfer. This setup facilitates the use of the transfer (Tx) line to enable communication with the Arduino, which is required for signaling the start and stop of power measurement during the cryptographic benchmarking process. This configuration ensures that the Arduino precisely calculates and tracks the energy consumption of the Raspberry Pi as it executes the selected LWCCs. More detailed information about this connection and its role in the overall benchmarking framework will be elaborated in the upcoming section on metrics.

3.4 Evaluation Metrics

The following metrics are used in our evaluation: throughput, CpB, code size, memory footprint, energy consumption, and *E-Rank*.

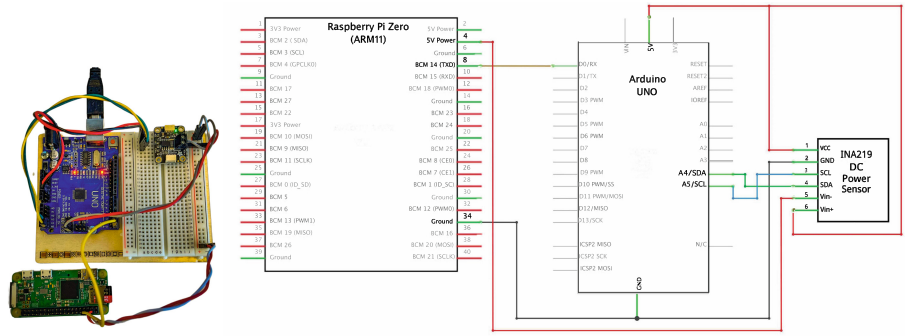
Throughput refers to the rate at which an LWCC can process data and is measured in kilobits per second (Kbps). Higher throughput values indicate rapid encryption or decryption speeds, offering a significant advantage for applications requiring swift data processing.

Cycles per Byte (CpB) indicates the average number of CPU cycles required to process one byte of data. Here, we used a specialized method to improve the precision of CpB measurements for LWCCs on the ARM11 processor. Our strategy involves direct access to the ARM timer registers, and the system timer registers. These registers are essential for managing various timing and scheduling operations within the Raspberry Pi's operating system. By interfacing directly with these hardware timers, our program can meticulously measure processor cycles by capturing timer values straight from the hardware. This method ensures a highly accurate assessment of the computational effort each LWCC demands, providing valuable insights into their efficiency on constrained devices.

To ensure accurate measurement of CpB for the selected LWCCs without being influenced by background processes, the system first calculates the CpB during a baseline period in which only background processes are running. This baseline CpB is then averaged over a specific time period. During the execution of LWCCs, the CpB is determined by subtracting the baseline CpB from the observed CpB at each instant. These differences are then summed to obtain the total processor cycles utilized during the cipher's execution. This approach improves measurement precision and mitigates the processor cycles of the background processes. Lower values of CpB indicate faster LWCC performance, requiring fewer processing cycles for encryption or decryption.

Code size refers to the amount of memory used in the non-volatile memory, which is the flash memory of the target device. In the case of the Raspberry Pi Zero W, the code size quantifies the storage space occupied by the LWCC, measured in bytes.

Memory footprint is the amount of memory used by a process during its execution. In our approach, we compute the memory footprint of LWCCs by assessing the volatile memory and subsequently dividing it by the size of the input. This computation yields the amount of volatile memory required (in bytes) for processing each byte of the data input. The benchmarking tool carefully monitors and records the amount of memory footprint of each selected LWCC implementation. The memory footprint assessment relies on the Resident Set Size (RSS), which represents the portion of memory reserved for a process in the main memory (RAM). In this context, RSS represents the portion of memory held by an LWCC within the RAM, excluding any data swapped out to disk. It contains memory allocated for important segments such as the code segment, data segment, heap, and stack, offering a comprehensive view of the actual memory usage of the LWCC. This provides valuable insights into the memory footprint of the ciphers and their impact on system resources.



(a) Physical Configuration (b) Schematic Diagram of the Connection Setup

Fig. 2. Connection setup between a Raspberry Pi and an Arduino UNO for power and energy measurement using an INA219 module.

Energy consumption refers to the quantity of power consumed by a process during its execution over a specified period of time. In this context, it refers to the energy consumption of LWCCs on the ARM11 processor, calculated in microjoules per byte (μJ/B). Due to limitations in directly measuring energy consumption via software, a hardware-based approach was devised to obtain precise power measurements. As illustrated in Fig. 2, the setup involves a circuit schematic specifically designed for power measurement and energy calculation. The Raspberry Pi board is powered by an Arduino UNO, which supplies power through GPIO pin-4 (5V power supply) and pin-6 (ground). A digital current and power monitor sensor developed by Texas Instruments, known as the INA219 sensor [25], is integrated into the power line from the Arduino to the Raspberry Pi. This sensor is capable of measuring the power consumption of Raspberry Pi with precision, enabling it to directly monitor and record electrical parameters.

During the initialization, the sensor monitors the power consumption of the Raspberry Pi while it executes background processes for a set duration. This data is then averaged to determine the typical power consumption during idle operation. Further, GPIO pin-14 on the Raspberry Pi is connected to pin-0 on the Arduino, establishing a one-way serial communication channel from the Raspberry Pi to the Arduino. This channel is required to signal the start and end of power and energy measurements. When encryption begins, the Raspberry Pi sends a signal to the Arduino, indicating the cipher's name along with its block and key size. The Arduino then activates the INA219 to start logging the power usage. Upon successful completion of the encryption/decryption process, another signal is transmitted to the Arduino to halt the power and energy measurement for that specific LWCC. To ensure that energy consumption is accurately calculated for the specific LWCC and not affected by background processes, the average power consumption calculated during the initialization is subtracted from the current power readings. The subtraction isolates the power used exclusively by the cipher during each discreet time interval. This approach mitigates the impact of background processes, thereby enhancing the precision and accuracy of energy consumption measurements for the LWCCs.

Rank measures the overall software efficiency of LWCCs and is defined as the ratio of the amount of data processed per second (throughput) to the aggregate amount of volatile and non-volatile memory used during the execution of cipher, as seen in Eq. 1. The overall software performance metric (*Rank*) was introduced by Beaulieu et al. [4], but in our work, we use a modified *Rank* metric (*E-Rank*) based on the Figure of Merit (FoM), which is used to evaluate the overall hardware performance. FoM is shown in Eq. 2 and was formulated by Badel et al. [3] to evaluate their LWBC called ARMADILLO. The authors highlighted the absence of power dissipation considerations in their FoM metric. This omission resulted from the knowledge that dynamic power, which is linked to the total switched capacitance, maintains a direct proportionality with area and, consequently, gate count. As a result, power is conventionally associated with the number of gates within the hardware context. As can be seen in Eq. 1, the *Rank* metric relies on aggregated memory utilization for calculation. Unlike FoM, memory utilization does not have direct proportionality with power dissipation due to no direct relation with the switched capacitance. Thus, integrating energy consumption into the *Rank* formulation would provide a comprehensive and more precise view of the overall software performance, referred to as *E-Rank*. Equation 3 provides the *E-Rank* formula, where accuracy is upheld through unit normalization, ensuring the uniform expression of all metrics in bytes.

$$Rank = \frac{\text{Throughput}}{\text{ROM} + 2 \times \text{RAM}} \tag{1}$$

$$FoM = \frac{\text{Throughput}}{\text{Clock freq} \times \text{Gate count}^2} \tag{2}$$

$$E\text{-}Rank = \frac{\text{Throughput}}{(\text{ROM} + 2 \times \text{RAM}) \times \text{Energy consumption}} \tag{3}$$

4 Experimental Results

4.1 Throughput Analysis

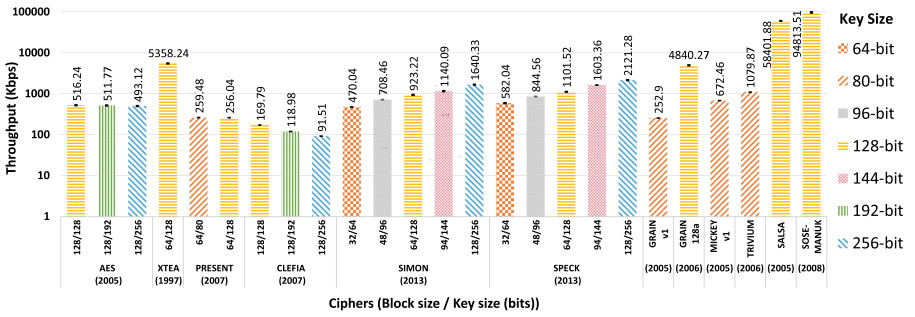


Fig. 3. Throughput of selected LWCCs.

Among the LWBCs, as depicted in Fig. 3, XTEA demonstrates the highest throughput, followed by SPECK and SIMON. SPECK is recognized as a software-oriented cipher and thus exhibits better throughput than SIMON. CLEFIA has a lower throughput compared to PRESENT, while AES has a higher throughput than both.

Regarding LWSCs, as demonstrated in Fig. 3, these ciphers collectively exhibit significantly better throughput than that of block ciphers, with the exception of Grain-v1. Sosemanuk leads the group with the highest throughput, followed by Salsa and then Grain-128a (optimized version). Trivium demonstrates better throughput compared to Mickey, followed by Grain-v1.

4.2 Cycles per Byte (CpB) Analysis

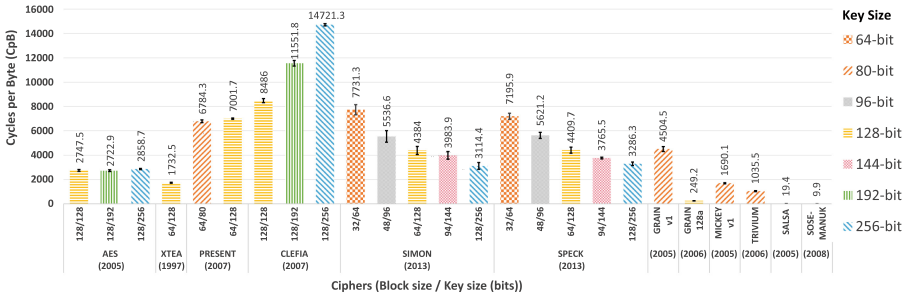
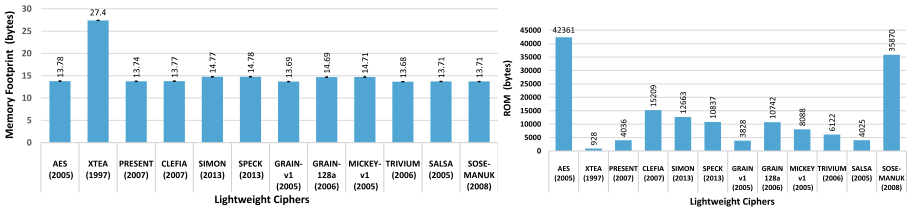


Fig. 4. Processor cycles of selected LWCCs.

Among LWBCs, as represented in Fig. 4, CLEFIA demonstrates the highest processor cycle count (CpB), followed by PRESENT. Contrarily, XTEA exhibits the lowest consumption, trailed by AES, SIMON, and SPECK. Note that the exception of 64-bit and 144-bit key sizes of SPECK show lower CpB than SIMON. AES displays lower CpB consumption compared to both SIMON and SPECK.

Regarding LWSCs, illustrated in Fig. 4, the overall processor cycle count is remarkably lower than that of LWBCs. Grain-v1 consumes the most among LWSCs, whereas Sosemanuk, Salsa, and Grain-128a (optimized version) exhibit successively lower CpB consumption. Trivium shows lower CpB utilization compared to Mickey.

4.3 Memory Consumption Analysis



(a) Memory footprint (b) ROM utilization

Fig. 5. Memory footprint and ROM utilization of selected LWCCs.

The memory footprint, as illustrated in Fig. 5a, demonstrates that XTEA consumes the highest memory footprint, followed by SPECK and SIMON among LWBCs. Contrarily, the lowest footprint is observed with PRESENT, CLEFIA, and AES. In the LWSCs, Mickey-v1 and Grain-128a exhibit the highest memory footprints, while Trivium, Grain-v1, Salsa, and Sosemanuk show successively lower footprints.

The analysis of code size, illustrated in Fig. 5, shows that XTEA boasts the smallest code size, while AES exhibits the largest among LWBCs. XTEA leads with the smallest code size, followed by PRESENT, SPECK, SIMON, and then CLEFIA in ascending order. In the LWSCs, Sosemanuk has the largest code size, while Grain-v1 has the smallest. The Grain-v1 is followed by Salsa, Trivium, Mickey, and Grain-128a in successive order of increasing code size.

4.4 Energy Consumption Analysis

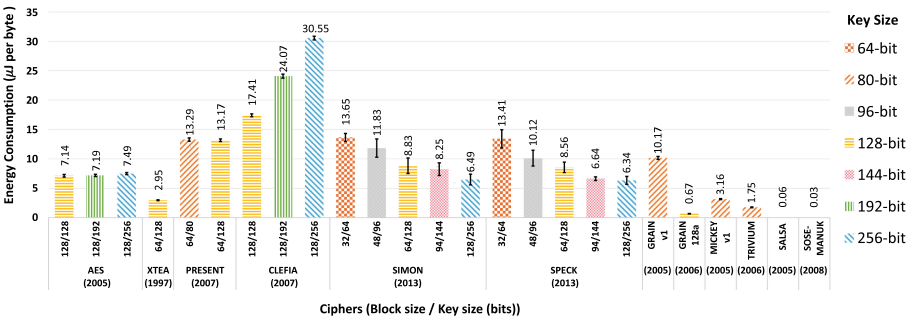


Fig. 6. Energy consumption of selected LWCCs.

Among LWBCs, as shown in Fig. 6, CLEFIA demonstrates the highest energy consumption, followed by PRESENT. Contrarily, XTEA exhibits the lowest energy consumption, followed by AES, SPECK, and SIMON. AES indicates lower energy consumption compared to both SIMON and SPECK, except for the 256-bit key size.

Among LWSCs, illustrated in Fig. 6, the overall energy consumption of LWSCs is remarkably lower than that of LWBCs. Grain-v1 consumes the most energy among LWSCs, while Sosemanuk, Salsa, and Grain-128a (optimized version) demonstrate successively lower energy consumption. Trivium shows lower energy consumption compared to Mickey.

4.5 E-Rank Analysis

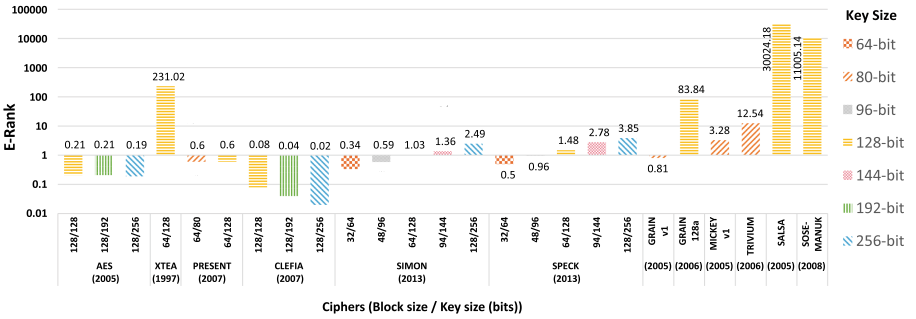


Fig. 7. E-Rank of selected LWCCs.

In our analysis of LWBCs, XTEA emerged as the best LWCC in overall software performance (*E-Rank*), as represented in Fig. 7, with CLEFIA demonstrating the least favorable results. XTEA's performance is followed by SPECK and then SIMON. Although AES displayed better throughput and energy consumption than PRESENT from our previous results, the *E-Rank* metric favored PRESENT, which is in line with its classification as a LWCC. Moreover, SPECK demonstrated better *E-Rank* utilization in comparison to SIMON, which indicates its orientation toward software implementation.

In the context of LWSCs, Fig. 7 shows that while Sosemanuk demonstrated substantial throughput and energy consumption, Salsa demonstrated better performance in terms of *E-Rank*, followed by Sosemanuk. On the other hand, Grain-v1 presented the least favorable *E-Rank*, while the optimized version of Grain-128a demonstrated better *E-Rank* than Mickey and Trivium. Trivium shows better overall software performance than Mickey.

5 Discussion

This study has systematically evaluated the performance and resource usage of LWCCs through a comprehensive benchmarking framework. Our analysis covered all the essential metrics, such as processor cycles, throughput, energy consumption, and total memory utilization, which includes both memory footprint and flash memory.

The performance measurement of LWBCs in Table 1 illustrates that XTEA demonstrates leading performance, exhibiting the best throughput, lowest CpB, and minimal energy consumption. This efficiency can be attributed to XTEA's simple and compact structure, which utilizes basic operations like XOR, AND, and SHIFT within a Feistel Network. This network splits the plaintext into two equal blocks, performing cryptographic operations on these divided segments. Following XTEA, SPECK and SIMON were notable for their throughput, with SPECK's performance surpassing SIMON's due to its Addition Rotation XOR (ARX) based structure, tailored for software optimization, compared to SIMON's hardware-oriented Feistel structure. PRESENT and CLEFIA, which are based on the Substitution Permutation Network (SPN) and Generalized Feistel Network (GFN), respectively, demonstrated inferior software performance compared to AES due to their hardware-oriented design. In the SPN design framework, block ciphers are created by applying multiple rounds of substitution and permutation to perform confusion and diffusion, respectively. Unlike Feistel Networks, the GFN splits the plaintext into sub-blocks of greater than or equal to $2n$ bits.

Among LWSCs, Sosemanuk and Salsa have shown remarkable throughput, lowest CpB, and energy efficiency. This is due to their software-oriented structures. Sosemanuk combines elements of synchronous stream and block ciphers, enhancing its initialization speed and overall performance, while Salsa employs a complex structure of addition modulo, bitwise XOR, and constant distance rotation, optimizing it for software implementation. On the contrary, hardware-oriented LWSCs like Grain, Mickey, and Trivium exhibited lower performance efficiency, except Grain-128a, which showed the best performance among them due to its speed-optimized implementation.

The analysis of resource usage among LWBCs reveals that XTEA, despite its best performance efficiency, demands the highest memory footprint, whereas AES requires the largest code size. Following closely, SIMON and SPECK demonstrate considerable memory usage while maintaining an optimal code size, achieving a balance between performance and resource consumption. PRESENT, on the other hand, is the most resource-efficient. Among LWSCs, Sosemanuk leads in resource usage in terms of code size, followed by Grain-128a and Mickey-v1. This analysis reveals a consistent trend: LWCCs that deliver higher performance efficiency also tend to require more resources, indicating a trade-off that necessitates careful cipher selection based on a balanced consideration of both performance and resource constraints to ensure optimal system functionality.

Integrating performance efficiency with resource usage is effectively accomplished through the *E-Rank* metric, which provides a comprehensive measure-

Table 1. Software implementation results of LWCCs on ARM processor, along with E-RANK

Ciphers	Block Length (bits)	Key Size (bits)	Code Size (bytes)	Memory Footprint (bytes)	Processor Cycles (CpB)	Throughput (Kbps)	Energy (μ J/B)	E-RANK
AES	128	128	42361	13.78	2747.5	516.24	7.14	0.21
		192			2722.9	511.77	7.19	0.21
		256			2858.7	493.12	7.49	0.19
XTEA	64	128	928	27.4	1732.5	5358.24	2.95	231.02
PRESENT	64	80	4036	13.74	6784.3	259.48	13.29	0.6
		128			7001.7	256.04	13.17	0.6
CLEFIA	128	128	15209	13.77	8486	169.79	17.41	0.08
		192			11551.8	118.98	24.07	0.04
		256			14721.3	91.51	30.55	0.02
SIMON	32	64	12663	14.77	7731.3	470.04	13.65	0.34
	48	96			5536.6	708.46	11.83	0.59
	64	128			4384	923.22	8.83	1.03
	96	144			3983.9	1140.09	8.25	1.36
	128	256			3114.4	1640.33	6.49	2.49
SPECK	32	64	10837	14.78	7195.9	582.04	13.41	0.5
	48	96			5621.2	844.56	10.12	0.96
	64	128			4409.7	1101.52	8.56	1.48
	96	144			3765.5	1603.36	6.64	2.78
	128	256			3286.3	2121.28	6.34	3.85
Grain-v1	–	80	3828	13.69	4504.5	252.9	10.17	0.81
Grain-128a	–	128	10742	14.69	249.2	4840.27	0.67	83.84
Mickey-v1	–	80	8088	14.71	1690.1	672.46	3.16	3.28
Trivium	–	80	6122	13.68	1035.5	1079.87	1.75	12.54
Salsa	–	128	4025	13.71	19.4	58401.88	0.06	30024.18
Sosemanuk	–	128	35870	13.71	9.9	94813.51	0.03	11005.14

ment of the overall software performance of LWCCs. As illustrated in Eq. 3, the *E-Rank* yielded several notable insights. In the category of LWBCs, XTEA achieved the highest *E-Rank* due to its balanced code size, memory footprint, and throughput, followed by SPECK, which surpassed SIMON. On the contrary, hardware-oriented ciphers such as SIMON, CLEFIA, and PRESENT demonstrated lower *E-Rank* values, with PRESENT outperforming AES due to its better resource management. In the selected set of LWSCs, software-oriented ciphers like Salsa achieved the highest *E-Rank* due to their efficient resource usage, despite Sosemanuk's better performance efficiency. Hardware-oriented LWSCs such as Grain-v1, Mickey, and Trivium are ranked lower, with Trivium showing comparatively better *E-Rank* due to more balanced performance and resource consumption. Additionally, Grain-128a, being an optimized version, naturally exhibited a higher *E-Rank* than Grain-v1, Mickey, and Trivium, highlighting the benefits of optimization in enhancing the cipher's overall software performance. So, *E-Rank* assists in identifying cryptographic solutions that offer an optimal balance between performance and resource efficiency, customized to the specific needs of resource-constrained environments.

6 Related Work

We started our investigation by examining various benchmarking tools designed to evaluate the performance of LWCC on resource-constrained devices. Although we considered several tools that offered certain performance metrics, there were specific limitations or functionality gaps in those tools that were important to our research objectives. We were motivated by the need for a comprehensive performance analysis covering all the major performance metrics like memory consumption, processor utilization, execution speed, and energy consumption. As a result, we decided to develop a new benchmarking framework from scratch rather than utilizing the existing platforms. The decision to create a new benchmarking framework was necessary due to various challenges posed by the existing frameworks, such as platform interdependency issues, code complexity, and inherent limitations to add new LWCCs.

In the BLOC project [11], a total of 16 LWBCs were evaluated using an MSP430 microcontroller. However, the accompanying C library lacked a standardized interface for all ciphers, causing integration difficulties with new platforms. It is important to note that the project only measured the performance of LWBCs in terms of memory and processor cycle count without providing any insights on energy consumption.

Dinu et al. [14] designed and developed a framework with the aim of achieving fair and consistent benchmarking of LWBCs across 8, 16, and 32-bit processors. This tool enables the analysis and comparison of the execution time, RAM requirements, and code size of lightweight primitives across various embedded platforms. The framework, while versatile, focused exclusively on block cipher primitives, providing performance information only in terms of memory. The associated research paper provides detailed documentation of the research methodology and findings.

The ECRYPT Benchmarking of Cryptographic Systems (eBACS) [15] is a part of the ECRYPT II project and measures cryptographic primitives' speed. It covers public-key systems (eBATS), stream ciphers (eSTREAM), and hash functions (eBASH). The project is built on the SUPERCOP (System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives) framework, which offers an extensive range of cryptographic primitive implementations. The framework's source code is open and written in C with inline assembly, Bash, and Python, making it easy to add new LWCC implementations across the research community. While it is valuable for speed assessment, it lacks any insights into energy and memory consumption.

7 Conclusions

In this paper, we introduced a benchmarking framework optimized for ARM processors, designed to evaluate the performance of selected LWCCs across multiple dimensions: processor speed, memory efficiency, and energy utilization. To accurately measure energy consumption, we enhanced the framework with a hardware extension comprising an Arduino and a power measurement sensor.

In the category of LWBCs, XTEA stood out by offering the lowest CpB, highest throughput, and lowest energy consumption, indicating faster execution. However, it also exhibited the highest total memory consumption among the LWBCs. Similarly, in the LWSCs, Sosemanuk demonstrated the lowest CpB, highest throughput, and lowest energy consumption, yet it also recorded the highest total memory consumption among LWSCs. These findings highlight the trade-offs between performance efficiency and resource usage that must be considered when selecting cryptographic solutions for resource-constrained environments. The *E-Rank* metric has demonstrated significant practical insights, effectively providing an optimum balance between performance, energy, and resource utilization. For instance, PRESENT performed better than AES in terms of *E-Rank* despite AES demonstrating efficient performance. This is because PRESENT has better resource management, which is an essential feature that is expected from LWCC. Similarly, among the selected LWSCs, Salsa ranked highest in overall software performance due to its more effective resource usage, even though Sosemanuk exhibited better performance efficiency. These findings validate the effectiveness of the *E-Rank* metric in providing an optimum and balanced evaluation of the overall software performance of LWCCs.

The future work involves measuring the LWCCs in real-world, resource-intensive applications, such as live video streaming via a camera module, and evaluating the performance of encryption and decryption processes within these applications.

Acknowledgments. This research received no external funding.

Disclosure of Interests. The authors have no conflicts of interest to declare.

References

1. ARM: ARM1176JZF-S Technical Reference Manual r0p7. ARM Limited (2009). https://developer.arm.com/documentation/ddi0301/h
2. Babbage, S., Dodd, M.: The MICKEY stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 191–209. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_15
3. Badel, S., et al.: Armadillo: a multi-purpose cryptographic primitive dedicated to hardware. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010. CHES 2010. LNCS, vol. 6225, pp. 398–412. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_2
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck block ciphers on avr 8-bit microcontrollers. In: Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014, Istanbul, Turkey, 1–2 September 2014, Revised Selected Papers 3, pp. 3–20. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-319-16363-5_1
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, pp. 1–6 (2015). https://doi.org/10.1145/2744769.2747946

6. Berbain, C., et al.: Sosemanuk, a fast software-oriented stream cipher. New Stream Cipher Designs: The eSTREAM Finalists, pp. 98–118 (2008). https://doi.org/10.1007/978-3-540-68351-3_9
7. Bernstein, D.J.: The salsa20 family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs: The eSTREAM Finalists, LNCS, vol. 4986, pp. 84–97. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_8
8. Birrell, E., Gjerdrum, A., van Renesse, R., Johansen, H., Johansen, D., Schneider, F.B.: Sgx enforcement of use-based privacy. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society, pp. 155–167 (2018). https://doi.org/10.1145/3267323.3268954
9. Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., Scavenius, O.: Rabbit: a new high-performance stream cipher. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 307–329. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39887-5_23
10. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
11. Cazorla, M., Marquet, K., Minier, M.: Survey and benchmark of lightweight block ciphers for wireless sensor networks. In: 2013 International Conference on Security and Cryptography (SECRYPT), pp. 1–6. IEEE (2013)
12. De Canniere, C., Preneel, B.: Trivium. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs: The eSTREAM Finalists, LNCS, vol. 4986, pp. 244–266. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
13. Dhanda, S.S., Singh, B., Jindal, P.: Lightweight cryptography: a solution to secure iot. Wirel. Pers. Commun. **112**(3), 1947–1980 (2020). https://doi.org/10.1007/s11277-020-07134-3
14. Dinu, D., Biryukov, A., Großschädl, J., Khovratovich, D., Le Corre, Y., Perrin, L.: Felics-fair evaluation of lightweight cryptographic systems. In: NIST Workshop on Lightweight Cryptography, vol. 128. NIST (2015)
15. Eisenbarth, T., et al.: Compact implementation and performance evaluation of block ciphers in ATtiny devices. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 172–187. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31410-0_11
16. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. Int. J. Wirel. Mob. Comput. **2**(1), 86–93 (2007). https://doi.org/10.1504/IJWMC.2007.013798
17. ISO Central Secretary: Information security – lightweight cryptography part 2: Block ciphers. Standard ISO/IEC 29192-2:2019, International Organization for Standardization, Geneva, CH (2019). https://www.iso.org/standard/78477.html
18. Martin, Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**(1), 48–59 (2011). https://doi.org/10.1504/IJWMC.2011.044106
19. Needham, R.M., Wheeler, D.J.: Tea extensions. Report (Cambridge University, Cambridge, UK, 1997) (1997)
20. Pettersen, R., Johansen, H.D., Johansen, D.: Secure edge computing with arm trustzone. In: IoTBDS, pp. 102–109 (2017). https://doi.org/10.5220/0006308601020109
21. Python Software Foundation: Python ctype Library Documentation (2024). https://docs.python.org/3/library/ctypes.html. Accessed 14 May 2024

22. Robshaw, M.: The eStream Project. In: New Stream Cipher Designs: The eSTREAM Finalists, pp. 1–6. Springer, London (2008). https://doi.org/10.1007/978-1-4471-5079-4_10
23. Saarinen, M.J.: Cryptanalysis of block tea, October 1998. Unpublished manuscript
24. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher clefia. In: Biryukov, A. (ed.) Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, 26-28 March 2007, Revised Selected Papers 14, FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74619-5_12
25. Texas Instruments: Ina219 zerø-drift, bidirectional current/power monitor with i2c interface (2021). https://www.ti.com/product/INA219
26. Wheeler, D.J., Needham, R.M.: Tea, a tiny encryption algorithm. In: Preneel, B. (eds.) Fast Software Encryption: Second International Workshop Leuven, Belgium, 14–16 December 1994, Proceedings 2, FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Berlin, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_29
27. Wu, H.: The stream cipher HC-128. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 39–47. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_4