ISSN: 2182-2069 / E-ISSN: 2182-2077

Comparative Performance Analysis of Lightweight Block Cipher Algorithms: From Single Core to Multicore

Yasamin Hamza Alagrash[1*], Salah Taha Allawi[2], and Jamal N. Hasoon[3]

[1*]Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq.
yhamza@uomustansiriyah.edu.iq, https://orcid.org/0000-0001-8400-1753

[2] Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq.
salah.taha@uomustansiriyah.edu.iq, https://orcid.org/0000-0002-5241-2504

[3] Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq.
jamal.hasoon@uomustansiriyah.edu.iq, https://orcid.org/0000-0003-0338-1200

Received: January 10, 2025; Revised: February 20, 2025; Accepted: April 04, 2025; Published: May 30, 2025

Abstract

Lightweight block encryption algorithms, such as PRESENT, KEILN, and LED, are pivotal in securing resource-constrained devices and small-scale systems relevance for environments with limited computational resources. This work presents a performance comparison in execution time, memory usage, and implementations in single-core and multicore systems. Our approach involves executing encryption for fixed text file sizes in single-mode and multicore-mode processing. The implemented Multicore technique is notable for its efficient adjustment of text files with a fixed size, showcasing its adaptability. The evaluation criteria encompass the execution time of encryption and memory utilization, continually showcasing the performance analysis of algorithms prioritizing lightweight encryption. These algorithms exhibit significant improvements while handling Multicore execution fixed text data files, which minimally impact encryption and decryption time. These values can be influenced by factors such as the encryption technique, file content, and system resources. We performed simulations using an Arduino Nano, which operates on an ATmega328P microcontroller as the principal device, and several microcontroller assessments within the IoT sector. This research additionally established additional measurements, covering energy utilization, security robustness, and attack resilience of three lightweight encryption methods. Our findings indicate that, in terms of various metrics, the LED approach appears to be the most effective algorithm in a single model. It has the quickest encryption time and uses the least amount of RAM.KEILN seems to be the best choice for applications where memory usage is critical because of its efficient use of resources. LEDs provide competitive speeds; however, a slight increase in memory usage is allowed. Because of its large memory requirements, PRESENT might not be as enticing even though it is reasonably quick. This paper will aid IoT developers in choosing a suitable platform and encryption method for creating a secure network, considering several resource constraint factors. It facilitates educated decision-making concerning algorithm choice, microcontroller selection, and optimization methodologies to attain the required equilibrium of performance, security, and resource utilization.

Journal of Interner Services and Information Security (JISIS), volume: 15, number: 2 (May), pp. 1-17.
DOI: *10.58346/JISIS.2025.I2.001*

*Corresponding author: Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq.

Keywords: Block Cipher, Lightweight Encryption Security, Multicore Encryption, Performance Evaluation of Lightweight Encryption.

1 Introduction

Cryptography is the practice that masks and keeps information confidential for protection from people who are not meant to have access to it and is intended only for specified people who can access it and share it securely. The problems of security are predicated upon these methods of mitigation by way of encryption cryptographic and authentication methods (Yang et al., 2020; Tello et al., 2023). In cryptography, a message generates coded information that changes messages by converting plain text into the output that entered user data and which will only perform the reverse decryption process back to original text (Mohammed & Abed, 2020).

The term 'Lightweight approach refers to a strategy that takes into account devices' speed, memory, and energy constraints. This approach is crucial in the Internet of Things (IoT), where devices are often resource-constrained (Dhanda et al., 2020). It necessitates the development of software, algorithms, and applications that can operate efficiently on such devices, thereby enhancing their performance(Abutaha et al., 2022; Ujjwal & Singh, 2024; Suryateja & Rao, 2024).

Lightweight cryptography (LWC) (Hameed et al., 2020). The expanding IoT sends a lot of critical information between machines and devices. The due concern of such information is to encrypt it so that it remains safe and confidential (Gupta & Kumar, 2021). Once again, encryption means changing (encrypting) data to prevent unauthorized access by people (Thakor et al., 2021). Most existing encryption algorithms and techniques were designed to enable secure connections between desktop computers and consumptive devices (Salman & Alomari, 2023). They thus do not work in resource-restricted devices such as those in IoT. (Goulart et al., 2022; Mhaibes et al., 2022). Sometimes classical encryption approaches may be far too slow and in turn reduce reaction time to also waste a lot of power. Thus one would require lightweight encryption approaches. The primary duty for lightweight encryption is to ensure data security between the sender and receiver in such a way that it accomplishes the process at a much cheaper cost and power consumption level than the existing non-lightweight encryption (Al-Husainy et al., 2021; Shetty et al., 2020).

This paper compares the implementation of lightweight block cipher algorithms: Klien, Present, and Lead, first in single-core and second in multicore. The evaluation criteria address the system's resources, such as memory storage and CPU time, with different IoT testbeds.

The rest of this paper is arranged as follows: Section Two explains the most recent related work, The Lightweight Cryptography as a Block Cipher, presented in Section 3; the proposed method is shown in Section 4; an evaluation of the proposed model is present in Section 5; and comparing it with previous work is discussed in Section 6 (Prakash & Samuel Ratna Kumar, 2016).

2 Related Work

Numerous studies have been conducted to assess cryptography algorithms in Internet of Things environments, with particular attention to resource constraints related to lifetime, storage, and performance (Jangra & Singh, 2019).

Hosseinzadeh & Bafghi, (2017) Conducted an assessment of the power consumption and memory utilization (Flash and RAM) of the SIMON, SPECK, TWINE, KLEIN, and Piccolo block ciphers on the Atmega128 microprocessor in a research article.

A study directed by (Singh & Deshpande, 2018) Compares the performance of six lightweight block ciphers for IoT devices (Suryateja & Rao 2025).

Jenny and her colleagues. (Panahi et al., 2021).This research study explores the efficiency of the Tiny Encryption Algorithm by analyzing the impact of different execution parameters. The study carefully examines the influence of variables such as the quantity of the data, the kind of processing, and the number of processing units in the execution machine on the performance of the tiny algorithm. By conducting this analysis, one can obtain significant insights that can be used to optimize (Al-hlalat, 2023; Jang et al., 2020).

In Radhakrishnan et al., (2024) compare the performance of three different LWC algorithms, AES-128, SPECK, and ASCON, based on single-core processing in implementation. The testbed uses an Arduino Nano board with other sensors.

In Fotovvat et al., (2021), a comparative study of the performance of 32 LWC algorithms in a single IoT node based on single-core processing with Raspberry Pi 3, Raspberry Pi Zero W, and iMX233. The main objective of this paper is to assess and prevent the use of lightweight symmetric ciphers on resource-limited devices by employing various strategies in both single-core and Multicore processing. An analysis was conducted on the block cipher implementations to assess their speed, cost, and efficiency in encrypting and decrypting various blocks (Goyal et al., 2013).

The analysis of lightweight cryptography algorithms is driven by determining the most efficient and safe algorithms suitable for resource-constrained devices, including IoT devices, mobile devices, and embedded systems. This enables the choice of the most suitable algorithm for a particular application and the optimization of its implementation to reduce resource consumption and enhance security (Wan & Hu, 2024; Hoa & Voznak, 2025). Additionally, it aids in detecting any flaws or susceptibilities in the algorithm that malicious individuals could potentially take advantage.

3 Lightweight Block Cipher

Lightweight cryptography as a block cipher is designed for low-power, low-cost devices like (IoT) end nodes and RFID tags. These devices can be implemented in hardware or software and use a variety of network protocols (Dewamuni et al., 2023; Shetty et al., 2021). Block ciphers are cryptographic algorithms that take into account factors such as block size, key size, operations, and key scheduling. The term "lightweight" refers to a cipher with a smaller block size (32, 48, or 64 bits) in comparison to a conventional cipher, which often has a larger block size (64 or 128 bits). Lightweight ciphers generally have smaller key sizes. In addition, lightweight ciphers have the potential to streamline the key schedule. (Rana et al., 2019) and utilize basic mathematical processes with more significant quantities (Ramadan et al., 2021). This section presents the three algorithms of block lightweight encryption, which are implemented and tested in our solution to demonstrate the quantitative definition. The primary rationale for selecting these three algorithms is their balance of security, efficiency, and appropriateness for contexts with constrained computational resources. They tackle the distinct issues such environments present, guaranteeing strong performance without depleting the available resources.

PRESENT Algorithm

P PRESENT is extremely lightweight cryptographic algorithms and thus presumes ideal choice for IoE systems. The PRESENT cipher belongs to the family of block cipher algorithms and it is based on Substitution-Permutation Network structure. The algorithm operates on a 64-bit block and can optionally use 80- or 128-bit keys. The plaintext is processed through several operations of the S-box layer, P-

layer, and XORed with Keys generated for 31 Rounds to obtain Ciphertext which again at the end of the Algorithm Operations XORed with the last generated key (Bharathi & Parvatham, 2022). In terms of key scheduling, the key is divided into four 32-bit words. A new round key into each round created by rotating the keywords and applying an S-box into one of the words (Thakor et al., 2021; Kubba & Hoomod, 2020).

KLEIN Algorithm

The algorithm was proposed by Zheng Gong and other researchers in 2012. It was engineered to be compatible with limited devices, such as wireless sensors and RFID tags. KLEIN is a set of block ciphers characterized by a block size of 64 bits and a key length that can vary between 64, 80, or 96 bits. The KLEIN cipher is denoted as KLEIN-64, KLEIN-80, or KLEIN-96, depending on the key length. The key length and block size are widely recognized as crucial elements in the trade-off between security and performance for a block cipher. The KLEIN algorithm utilizes a substitution-permutation network (SPN) structure, which involves substitution using 4-bit S-boxes, linear transformation based on the AES MixColumn operation, and key addition by XORing with round keys.

LED Algorithm

Light-emitting diode (LED) The algorithm was proposed by Guo et al. in 2011. The LED cipher is a cryptographic algorithm that operates on blocks of 64 bits. The key used in the LED cipher can have a size ranging from 64 to 128 bits. The essence of the cipher lies in a sequence of iterations, each comprising basic operations known as Substitution utilizing the 4-bit S-box, Permutation of bits, and a round of key manipulation. For 64-bit keys, the key is utilized as the initial round key without any modifications. For 128 bits in length, the key is split into two halves, each consisting of 64 bits (Raza et al., 2020). The first two round keys are then obtained from these halves. Generating subsequent round keys involves shifting and performing an XOR operation with the preceding round key (Mhaouch et al., 2023).

Figure 1 illustrates the comparison of three lightweight block cipher methods, emphasizing that these three algorithms target resource construction devices as their target platform, which can be considered as the main purpose of choosing these algorithms in this study. Table 1 shows the features of these three algorithms in resource-constrained environments.

Table 1: Objective of Present, Klein and Led lightweight Encryption Algorithms

Algorithm	Purpose	Use case
PRESENT	adjusted to accommodate varying resource limitations, rendering it suitable for various IoT devices, from diminutive sensors to more intricate smart devices.	Employed in industrial control systems where low power and minimal processing capabilities are critical.
KLEIN	ability to optimize AI solutions within resource-constrained environments	Edge computing devices
LED	adapted to accommodate varying degrees of resource limitations, rendering it appropriate for a diverse array of IoT devices, ranging from basic sensors to more sophisticated smart	IoT sensors

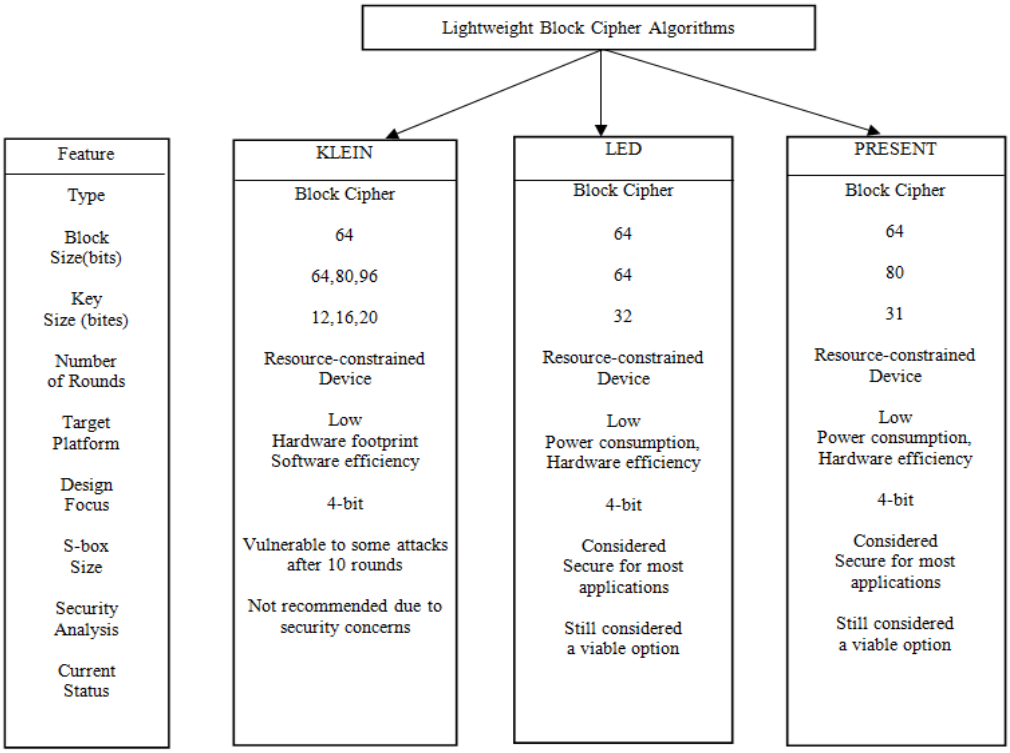

Figure 1: Lightweight Block Cipher Algorithms

4 Method of Comparison

This section provides the proposed approach for assessing the efficiency of light block cipher encryption methods. To be more precise, the evaluation and analysis of the three-block cipher algorithms will be conducted in two distinct manners: single-core cryptography and multicore cryptography. The main goal of this study is to evaluate the encryption and decryption methods for each algorithm using two different processing approaches: single-core and multicore. These methods are implemented using threads in both operations.

By implementing these modifications, the proposed system effectively minimizes waiting time during thread preparation, streamlines result handling and optimizes resource allocation. This results in enhanced efficiency and performance of the encryption and decryption processes, ultimately improving an algorithm's overall performance. The algorithm multicore crypto explains the Multicore crypto methodology.

Algorithm Multicore crypto	
Input: plain file Output: cipher file	
1.	Begin.
2.	Start multicore function.
3.	Identify the number of blocks.
4.	Create a block of data.
5.	Initial result storage.
6.	Defined encryption- decryption function
7.	Employs the lightweight encryption-decryption function to encrypt- and decrypt each block separately.
8.	End.

The evaluation algorithm is proposed to evaluate

Evaluation Algorithm
Input: cipher algorithm Output: compute run time, memory size

1. Evaluate execution time by using tic and toc functions //Toc, which reads the time that has passed since the tic function was called. tic is a function that initiates a timer. //
tic;
parfor i = 1:N
encryptedBlocks{i} = lightweightEncrypt (dataBlocks{i}, key);
end
toc;
2. Evaluate memory usage
memoryUsage = evaluateMemoryUsage (data, key, numCores);
printf ('Memory usage: %d bytes\n', memoryUsage);
3. End

5 Experiments and Results

In this section, the outcomes of the proposed evaluation approach are meticulously analysed and scrutinized. Various performance metrics are assessed, including encryption and decryption speeds, throughput, and resource utilization. These results are compared against baseline measurements to ascertain the efficacy of the evaluation methodology. Additionally, the analysis experiments are proposed based on the following criteria:

Three lightweight algorithms implement the single-core and multi-core approaches for file types, such as text for encryption and decryption. Several performance matrices are addressed for three lightweight algorithms: PRESENT, KEILN, and LED.

Compute the execution for encryption algorithms based on the following formula.

Execution Time = Tend- Sstart : ……(1)

Where Tend end time of encryption| decryption, decryption process, and Ts start time of encryption| decryption.

Memory Usage = Size Encrypted Data + Size Key + Size Round Variables …….. (2)

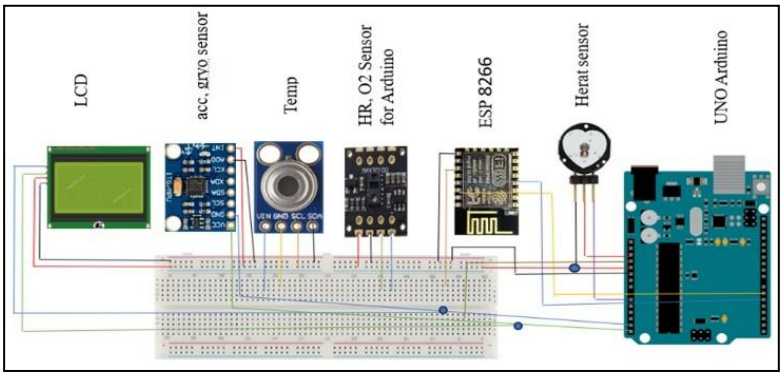

Figure 2: Experiments Testbed

As shown in Figure 2, the test-bed configuration is illustrated in the figure, particularly by the Arduino Nano, which operates on an ATmega328P microcontroller including 32 KB of flash memory

and 2 KB of RAM, as well as the Arduino Micro, which utilizes an ATmega32U4 microprocessor with 32 KB of memory and 2.55 KB of RAM. This board is considered due to their widespread adoption in the IoT sector and their commonality among various resource-constrained IoT devices, hence offering a solid foundation for evaluating the effectiveness and security of lightweight cryptographic techniques. The Raspberry Pi RP2040 microcontroller is equipped with twin cores, namely the 32-bit Arm Cortex-M0+ architecture. This enables the concurrent execution of many tasks.

Furthermore, a heartbeat sensor measures heart rate (the number of pulses per minute). Its function relies on quantifying alterations in blood flow and can be utilized in several health and athletic contexts to assess cardiac activity.

The NodeMCU panel ESP8266, featuring wireless connectivity, is the most prevalent controller in IoT applications. It can be programmed independently using Arduino software or connected to Arduino to function as a chip. It is utilized in numerous applications requiring Internet access.

The HR, O2 Sensor for Arduino is a dual sensor that measures heart rate (HR) and blood oxygen saturation (O2). This sensor is frequently employed in medical and fitness applications, requiring meticulous monitoring of the body's critical processes.

The Temperature Sensor with Arduino measures environmental or device temperature, and it is frequently employed in control and monitoring projects. Numerous temperature sensors are compatible with Arduino, each employing distinct methodologies for temperature measurement.

An accelerometer is a sensor that quantifies linear acceleration. Wearable devices frequently utilize it to assess movement or vibration and detect falls. Table 1 displays the outcomes of implementing the PRESENT algorithm on a single-core processor in time consumption and speed conditions.

Table 2: Encryption/Decryption Time in Single Core by Using PRESENT Algorithm

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.7704082	0.696345
Text2	0.7254006	0.5730557
Text3	0.7410101	0.6672154
Text4	0.8669648	0.5815883
Text5	0.7725059	0.5904124

Across all scenarios, the disparity in time between encryption and decryption is consistently minimal, with decryption consistently being quicker. Table 2 depicts the memory utilization associated with the encryption and decryption of text files, specifically in the context of single-core computing.

Table 3: Memory Usage in Single Core Implementation of PRESENT Algorithm

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	6828782	6828245
Text2	6828245	6828171
Text3	6823657	6827205
Text4	6827106	6827086
Text5	6826975	6825525

When processing all text files, the memory used for encryption is consistently greater than the memory used for decryption. Memory Variability: Memory utilization for encryption and decryption slightly differs among the various text files. Table 3 provides a detailed breakdown of the time required for the encryption and decryption process utilizing the PRESENT method in the Multicore approach.

Table 4: Implantation Results of Multicore-core of Present Algorithm

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.5683843	0.4608792
Text2	0.4900049	0.4077416
Text3	0.5279247	0.3771127
Text4	0.5030452	0.3868465
Text5	0.5213849	0.3878417

Similar to the results obtained for single-core processing, the findings for Multicore processing demonstrate that decryption is more efficient. Table 4 depicts the memory utilization associated with the encryption and decryption of text files, specifically in the context of multicore-core computing.

Table 5: Memory Usage in Multicore Implementation of PRESENT Algorithm

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	5992550	5981118
Text2	5982265	5986836
Text3	5946462	5949792
Text4	5936206	5947621
Text5	5947261	5947159

According to the table, encryption uses more memory than decryption for the specified text files. This implies that the encryption procedure may use more memory than decryption. To execute a KLEIN algorithm in single-core mode, the time spent on encryption and decryption stages is presented in Table 5.

Table 6: KLEIN Algorithm Implements in Single Core /Time Consuming

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.738463833	0.694533048
Text2	0.713749534	0.532610957
Text3	0.729951035	0.636536806
Text4	0.861905435	0.526721478
Text5	0.70385882	0.537556462

The encryption times range from approximately 0.70 to 0.86 seconds. There is some variation between the files, with Text4 taking the longest to encrypt and Text5 taking the shortest. The decryption times are generally lower than the encryption times, ranging from about 0.70 to 0.86 seconds. Similar to encryption, there is some variation between files, with Text2 being the fastest to decrypt and Text3 being the slowest. Table 6 illustrates the outcomes of the KLEIN implementation in terms of memory use matrices of single-core.

Table 7: Outcomes of the KLEIN Implementation Memory Use of Single-core

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	6304283	6342015
Text2	6601745	6514969
Text3	6152243	6722390
Text4	6599048	6578987
Text5	6416816	6777919

Overall, the memory utilization for encryption and decryption appears to be similar for each file, with little variations. Without further analysis, it is challenging to ascertain a consistent pattern about which process consumes more memory. Table 7 presents the time spent on encryption and decryption stages to execute a KLEIN algorithm in multicore-core mode.

Table 8: KLEIN Algorithm Implements in Multicore-Core/Time-Consuming

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.512418296	0.415083066
Text2	0.435014099	0.376867862
Text3	0.487993094	0.365612133
Text4	0.451200307	0.356554672
Text5	0.455895267	0.342347929

The encryption algorithm employed could be computationally costlier than the decoding algorithm. Table 8 illustrates the KLEIN implementation outcomes regarding multicore memory use matrices.

Table 9: Outcomes of the KLEIN Implementation Memory Use of Multicore Processing

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	5375440	5483339
Text2	5417176	5764173
Text3	5364168	5912803
Text4	5475910	5436098
Text5	5792673	5854606

Memory consumption for both encryption and decryption differ among files. There is no direct relationship between file names or any intrinsic sequence and the memory consumption figures. Typically, the amount of memory required for decryption is greater than that needed for encryption for each file. The ultimate implementation used an LED Algorithm that initially operated in a solitary mode and was assessed as time-consuming, as seen in Table 9. Which provides a preliminary look at five text files' encryption and decryption times.

Table 10: The LED Algorithm Implemented in Single-core/ Time-consuming

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.762254294	0.642588162
Text2	0.697840826	0.570678477
Text3	0.696628954	0.635553671
Text4	0.795188396	0.567046482
Text5	0.708271595	0.549075066

Table 10 depicts the memory utilization associated with the encryption and decryption of text files, specifically in the context of multicore-core computing.

Table 11: Outcomes of the LED Implementation Memory Use of Single-core

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	6525607	6454697
Text2	6202991	6665403
Text3	6476868	6169169
Text4	6180409	6271606
Text5	6540192	6473112

The disparity in memory use between each file's encryption and decryption processes is relatively insignificant. The ultimate implementation used an LED Algorithm that initially operated in a solitary mode and was assessed as time-consuming, as seen in Table 11 for multicore-core.

Table 12: The LED Algorithm Implemented in Multicore-core/ Time-consuming

File name 1024 bytes	Encryption Time(sec)/ single core	Decryption Time(sec)/ single core
Text1	0.564352875	0.451189076
Text2	0.482440635	0.366548866
Text3	0.476312601	0.328921032
Text4	0.451766307	0.344765358
Text5	0.453223733	0.332821241

File Sizes: All files are 1024 bytes in size. Encryption Time: Text1 has the longest encryption time, taking 0.564352875 seconds. Text5 has the shortest encryption time, taking only 0.451766307 seconds. The decryption time for Text1 is the highest, taking 0.451189076 seconds. Text3 has the shortest decryption time, clocking in at 0.328921032 seconds.

Table 12 depicts the memory utilization associated with the encryption and decryption of text files, specifically in the context of multicore computing.

Table 13: Outcomes of the LED Implementation Memory Use of Multicore

File name 1024 byte	Encryption memory usage in KB	Decryption memory usage in KB
Text1	5351903	5795693
Text2	5142252	5350115
Text3	5330821	5409996
Text4	5144575	5912665
Text5	5919729	5785061

Typically, the process of decrypting these files necessitates a somewhat greater amount of RAM compared to encrypting them in table 13.

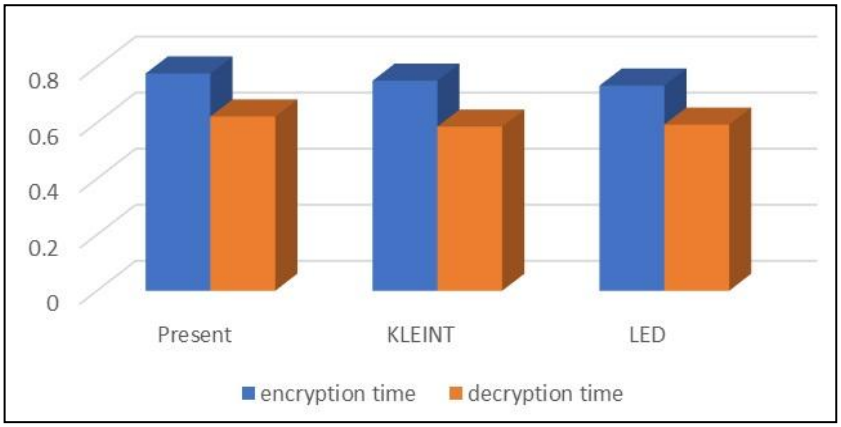

Figure 3: Time-consuming: Compares Algorithms Executed in Single Mode

Figure 3 compares algorithms in single mode and demonstrates the quicker, lightweight algorithm's superior performance in single-core processing, considering the fixed file size.

The present algorithm exhibits the shortest decryption time but the longest encryption time. KLEINT and LED exhibit comparable encryption and decryption speeds, with KLEINT demonstrating a minor advantage in encryption and LED showing a slight advantage in decryption.

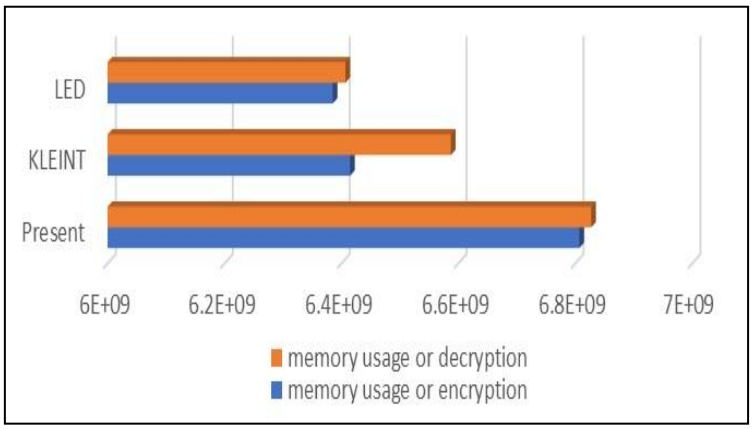

Figure 4: Memory usage in single mode

Efficient memory storage is achieved with an algorithm shown in Figure 4. The "Present" category typically exhibits more memory utilization or values than the "LED" category for orange and blue bars. The disparity in memory utilization between the "LED" and "Present" is more noticeable for the orange bars than the blue bars.

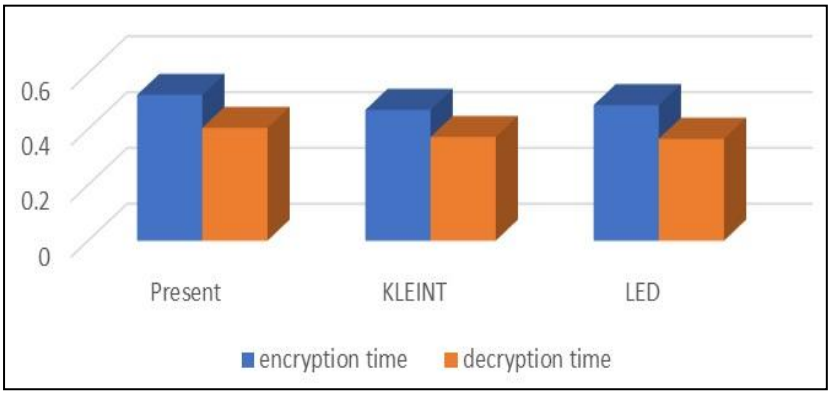

Figure 5: Performance Algorithms in Multicore

Figure 5 illustrates the Multicore performance with fixed file sizes for each algorithm. Addition Figure 6. Explain the memory usage of three algorithms in terms of encryption and decryption.

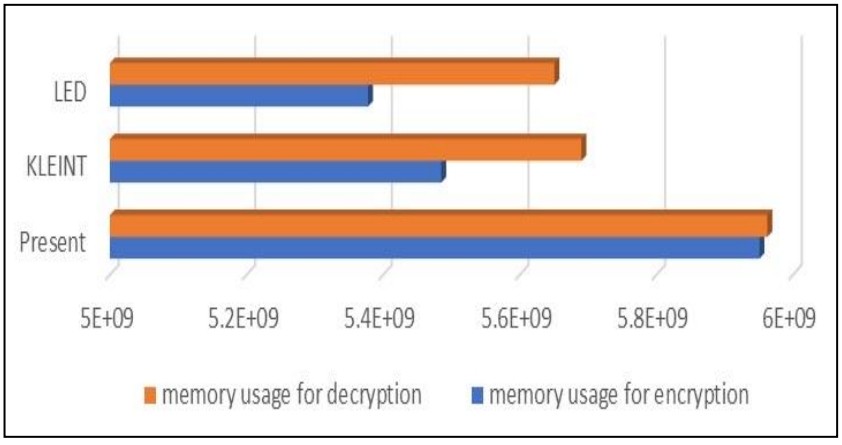

Figure 6: Memory usage in Multicore for 3 algorithms

6 Analysis and Discussion

This section of the paper provides an in-depth analysis that considers various metrics used to evaluate cryptographic algorithms.

Table 14: Shows Encryption and Decryption Performance Across Different Microcontrollers

Algorithm	Microcontroller	Core type	Encryption Time (ms)	Decryption Time (ms)	Memory Usage (KB)
PRESENT	ATmega328P (Arduino Nano)	8-bit AVR	750	620	6.8
PRESENT	ESP32	32-bit Dual-Core	480	400	5.9
PRESENT	Raspberry Pi Pico	32-bit Dual-Core	510	420	6.0
PRESENT	STM32F4	32-bit ARM Cortex-M4	470	390	5.8
KLEIN	ATmega328P (Arduino Nano)	8-bit AVR	730	600	6.5
KLEIN	ESP32	32-bit Dual-Core	460	380	5.5
LED	ATmega328P (Arduino Nano)	8-bit AVR	710	590	6.4
LED	ESP32	32-bit Dual-Core	450	370	5.3

Table 14 illustrates that the time needed for encryption and decryption varies significantly across different microcontrollers. As expected, the more powerful microcontrollers (ESP32, Raspberry Pi Pico, and STM32F4) generally exhibit faster encryption and decryption times than the less proficient ATmega328P. This highlights the impact of computational power on cryptographic effectiveness. In the algorithm, LED demonstrates somewhat improved performance relative to KLEIN and PRESENT in most cases, while the differences are not substantial. Memory use remains consistently stable across the algorithms and microcontrollers, displaying only minor variations. Encryption and decryption durations: Generally, decryption timings are slightly shorter than encryption periods for the same method and microcontroller.

In the case of e-energy consumption, The energy expended by a processor when executing a software program, such as a block cipher, is equivalent to the product of the average power dissipation and the entire execution time.

To measure energy consumption, the INA219 Power Sensor (measures voltage/current drawn), Arduino Energy Profiling, and ESP32 ADC Sensor were used to read power consumption. The energy

Consumption is based on the following calculation

$$E=P\times T \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where E is energy (joules), P is power (watts), T is execution time (seconds).

Table 15 presents a comprehensive overview of Present, Klein, and Led lightweight encryption algorithms in terms of various criteria. In this case, we consider a Brute-force attack a specific attack scenario in conducting thorough security evaluations.

Table 15: Comprehensive Evaluation of Algorithms

Algorithm	Execution time(ms)	Memory Usage (KB)	Energy Consumption (mJ)	Security Strength	Resistance to Attacks
PRESENT	750	6.8	0.15	Moderate	High
KLEIN	730	6.5	0.13	Low	Medium
LED	710	6.4	0.12	High	High

Furthermore, Table 15 explores the trade-offs among several performance dimensions. Although LED demonstrates superiority in speed, energy economy, and security, KLEIN may be used when memory utilization is paramount. LED is the optimal choice for high-security applications with constrained resources. KLEIN may be an appropriate option when performance is not paramount, and resources are limited.

Table 16 show the average execution for every algorithm in single and multicore in run time and memory usage.

Table 16: Average Performance Analysis: PRESENT, KEILN, LED of Single Core

Algorithm		Run time (sec)	Memory usage (KB)
PRESENT	Encryption	0.7752579	6826953.11
	Decryption	0.6217233	6827246.38
KEILN	Encryption	0.7495857	6414827.15
	Decryption	0.5855917	6587256.07
LED	Encryption	0.7320368	6385213.55
	Decryption	0.5929883	6406797.33

LED seems to be the most efficient algorithm in a single model in terms of runtime and memory utilization. It consumes the least amount of RAM and has the fastest encryption time. Performance Balance: KEILN is a decent trade-off between speed and resource consumption since it provides balanced performance with moderate run times and memory utilization. High Memory Usage: Despite its effectiveness, PRESENT uses the most memory and runs the longest. This may be a factor for systems with constrained resources.

Table 17: Average Performance Analysis: PRESENT, KEILN, LED of Multicore

Algorithm		Run time (sec)	Memory usage (KB)
PRESENT	Encryption	0.5221488	5960948941
	Decryption	0.4040843	5962505421
KEILN	Encryption	0.4685042	5485073.27
	Decryption	0.3712931	5690203.6
LED	Encryption	0.4856192	5377855979
	Decryption	0.3648491	5650706182

Because KEILN uses resources efficiently, it appears to be the ideal option for applications where memory utilization is crucial. LED gives competitive speeds, but if a little increasing memory utilization is acceptable. Despite being somewhat fast, PRESENT may be less appealing because of its high memory requirements in Table 17.

Nevertheless, using many threads brings up additional costs, especially during preparing the threads. Our proposed system can solve these issues by using the multicore function, which then handles the distribution of work among threads based on the available cores. This approach optimizes resource utilization and reduces waiting time by efficiently utilizing the available processing power in the absence of further information; it is necessary to ascertain the factors that account for the variations in memory utilization between files or between the encryption and decryption processes. Various factors, including the encryption algorithm, file content, and system resources, can all impact these values.

After evaluating the many measures employed to assess cryptographic algorithms, it is crucial to acknowledge that distinct algorithms may excel in different dimensions.

To the best of our knowledge, this study proposes and develops a method for evaluating the performance of block cipher encryption, specifically for lightweight ciphers, in the context of resource

constraints in the Multicore process. This method can improve performance and enhance system functionality.

For instance, consider that the encryption time for fixed-size files is relatively the same. The decryption times are generally lower than the encryption times. The content of the text files may influence the encryption/decryption times. Factors such as the encryption algorithm, file content, and system resources can all influence these values.

7 Conclusion

Evaluation and benchmarking have given clear indications of suitability of lightweight cryptography algorithms for resource-constrained IoT devices. LED fits best for implementations with minimum memory considerations while the cryptographic community has already analytically reviewed KEILN. All these operations consume much processing power. This could be expensive due to the CPU limitations of the targeted resource-constrained IoT devices.

PRESENT, KEILN, and LED. The performance analysis approach is based on encryption and decryption times across fixed data sizes by leveraging multicore cores for computation instead of relying solely on a single core. The algorithm covers fixed data sizes, ensuring comprehensive applicability. Evaluation metrics such as encryption time memory usage and energy consumption attest to the algorithm's efficiency. Experimental results highlight the superiority of memory usage. Moreover, even for a short time, it is significant. Multicore systems can markedly enhance the speed of encryption and decryption operations, even with lightweight methods. Complementary Strengths: The integration of lightweight encryption and Multicore processing offers an effective equilibrium of security and performance. For future work, the suggestion lies in the enhanced hybrid encryption algorithm (PRESENT-LED or KLEIN-PRESENT) for future advancements in encryption and decryption velocity, reduced memory usage, and enhanced energy efficiency.

Acknowledgment

The authors thank Mustansiriyah University (www.uomustansiriyah.edu.iq) in Baghdad, Iraq, for supporting this work.

References

[1] Abutaha, M., Atawneh, B., Hammouri, L. & Kaddoum, G. (2022). Secure lightweight cryptosystem for IoT and pervasive computing. *Scientific Reports, 12*(1), 1–15. https://doi.org/10.1038/s41598-022-20373-7

[2] Al-hlalat, M. (2023). Exploring the Tiny Encryption Algorithm: A Comparative Analysis of Parallel and Sequential Computation. *IJSDR, 8*(7), 693–702.

[3] Al-Husainy, M. A. F., Al-Shargabi, B. & Aljawarneh, S. (2021). Lightweight cryptography system for IoT devices using DNA. *Computers and Electrical Engineering, 95,* 107418. https://doi.org/10.1016/j.compeleceng.2021.107418

[4] Bharathi, R. & Parvatham, N. (2022). Light-Weight Present Block Cipher Model for IoT Security on FPGA. *Intelligent Automation and Soft Computing, 33*(1), 35–49. https://doi.org/10.32604/iasc.2022.020681

[5] Dewamuni, Z., Shanmugam, B., Azam, S. & Thennadil, S. (2023). Bibliometric Analysis of IoT Lightweight Cryptography. *Information (Switzerland), 14*(12). https://doi.org/10.3390/info14120635

[6] Dhanda, S. S., Singh, B. & Jindal, P. (2020). Lightweight Cryptography: A Solution to Secure IoT. In Wireless Personal Communications. *Springer US. 112* (3). https://doi.org/10.1007/s11277-020-07134-3

[7] Goulart, A., Chennamaneni, A., Torre, D., Hur, B. & Al-Aboosi, F. Y. (2022). On Wide-Area IoT Networks, Lightweight Security and Their Applications—A Practical Review. *Electronics (Switzerland), 11*(11). https://doi.org/10.3390/electronics11111762

[8] Goyal, D., Hemrajani, N., & Paliwal, K. (2013). GPH algorithm: Improved CBC improved BIFID cipher symmetric key algorithm. *International Journal of Communication and Computer Technologies, 1*(2), 83-86. https://doi.org/10.31838/IJCCTS/01.02.03

[9] Gupta, D. N. & Kumar, R. (2021). Sponge based Lightweight Cryptographic Hash Functions for IoT Applications. 2021 International Conference on Intelligent Technologies, *CONIT*. https://doi.org/10.1109/CONIT51480.2021.9498572

[10] Hameed, R. S., Hussein, A., Khalaf, B. A., Fadel, A. H., Hasoon, J. N., Mostafa, S. A. & Khalaf, A. (2020). A Light-weight ESalsa20 Ciphering based on 1D Logistic and Chebyshev Chaotic Maps. *Solid State Technology*. *63*(1), 1078-1093. https://www.researchgate.net/publication/344492787

[11] Hoa, N. T., & Voznak, M. (2025). Critical review on understanding cyber security threats. *Innovative Reviews in Engineering and Science, 2*(2), 17-24. https://doi.org/10.31838/INES/02.02.03

[12] Hosseinzadeh, J. & Bafghi, A. G. (2017). Software Implementation and Evaluation of Lightweight Symmetric Block Ciphers of The Energy Perspectives and Memory. *International Journal of Engineering Education /(IJEE), 9*(2), 1–6. https://doi.org/10.48550/arXiv.1706.03909

[13] Jang, K., Song, G., Kim, H., Kwon, H., Kim, H. & Seo, H. (2021). Efficient implementation of present and gift on quantum computers. *Applied Sciences (Switzerland), 11*(11). https://doi.org/10.3390/app11114776

[14] Jangra, M. & Singh, B. (2019). Performance analysis of CLEFIA and PRESENT lightweight block ciphers. *Journal of Discrete Mathematical Sciences and Cryptography, 22*(8), 1489–1499. https://doi.org/10.1080/09720529.2019.1695900

[15] Kubba, Z. M. J. & Hoomod, H. K. (2020). Developing a lightweight cryptographic algorithm based on DNA computing. *AIP Conference Proceedings, 2290*. https://doi.org/10.1063/5.0027361

[16] Mhaibes, H. I., Abood, M. H. & Farhan, A. K. (2022). Simple Lightweight Cryptographic Algorithm to Secure Imbedded IoT Devices. *International Journal of Interactive Mobile Technologies, 16*(20), 98–113. https://doi.org/10.3991/ijim.v16i20.34505

[17] Mhaouch, A., Elhamzi, W., Abdelali, A. Ben & Atri, M. (2023). Efficient Design for a Hardware Implementation of the LED Block Cipher. *Journal Europeen Des Systemes Automatises, 56*(5), 725–733. https://doi.org/10.18280/jesa.560502

[18] Mohammed, M. A. & Abed, F. S. (2020). Cloud Storage Protection Scheme Based on Fully Homomorphic Encryption. *Aro-the Scientific Journal of Koya University, 8*(2), 40–47. https://doi.org/10.14500/aro.10590

[19] Panahi, P., Bayılmış, C., Çavuşoğlu, U. & Kaçar, S. (2021). Performance Evaluation of Lightweight Encryption Algorithms for IoT-Based Applications. *Arabian Journal for Science and Engineering, 46*(4), 4015–4037. https://doi.org/10.1007/s13369-021-05358-4

[20] Prakash, S., & Samuel Ratna Kumar, P. S. (2016). Productivity Enhancement in Steering Knuckle Line. *International Journal of Advances in Engineering and Emerging Technology, 7*(1), 130–137.

[21] Ramadan, R. A., Aboshosha, B. W., Yadav, K., Alseadoon, I. M., Kashout, M. J. & Elhoseny, M. (2021). LBC-IoT: Lightweight Block Cipher for IoT Constraint Devices. *Computers, Materials and Continua, 67*(3), 3563–3579. https://doi.org/10.32604/cmc.2021.015519

[22] Rana, S., Anwar Hussen Wadud, M., Azgar, A. & Abul Kashem, M. (2019). A Survey Paper of Lightweight Block Ciphers Based on Their Different Design Architectures and Performance Metrics. *International Journal of Computer Engineering and Information Technology, 11*(6), 119–129. www.ijceit.org

[23] Rashidi, B. (2020). Flexible structures of lightweight block ciphers PRESENT, SIMON and LED. IET Circuits, *Devices and Systems, 14*(3), 369–380. https://doi.org/10.1049/iet-cds.2019.0363

[24] Raza, A. R., Mahmood, K., Amjad, M. F., Abbas, H. & Afzal, M. (2020). On the efficiency of software implementations of lightweight block ciphers from the perspective of programming languages. *Future Generation Computer Systems, 104*, 43–59. https://doi.org/10.1016/j.future.2019.09.058

[25] Salman, R. H., & Alomari, E. S. (2023). Survey: Homomorphic Encryption-based Deep Learning that Preserves Privacy. *International Academic Journal of Science and Engineering, 10*(2), 153–163. https://doi.org/10.9756/IAJSE/V10I2/IAJSE1019

[26] Sherine Jenny, R., Sudhakar, R. & Karthikpriya, M. (2021). Design of Compact S Box for Resource Constrained Applications. *Journal of Physics: Conference Series, 1767*(1). https://doi.org/10.1088/1742-6596/1767/1/012059

[27] Shetty, V. S., Anusha, R., MJ, D. K., & Hegde, P. (2020, February). A survey on performance analysis of block cipher algorithms. In *2020 International Conference on Inventive Computation Technologies (ICICT)* (pp. 167-174). IEEE. https://doi.org/10.1109/ICICT48043.2020.9112491

[28] Singh, P., & Deshpande, K. (2018). Performance evaluation of cryptographic ciphers on IoT devices. https://doi.org/10.48550/arXiv.1812.02220

[29] Suryateja, P. S. & Rao, K. V. (2024). A Survey on Lightweight Cryptographic Algorithms in IoT. *Cybernetics and Information Technologies, 24*(1), 21–34. https://doi.org/10.2478/cait-2024-0002

[30] Suryateja, P. S., & Rao, K. V. (2025). Performance Evaluation of Contemporary Block Ciphers for IoT Applications. *Journal of Internet Services and Information Security, 15(1),* 16-31. https://doi.org/10.58346/JISIS.2025.I1.002

[31] Tello, A. B., Alam, S., Salve, A. R., Kumari, B. M. K., & Arora, M. (2023). Quantitative Evaluation of Android Application Privacy Security Based on Privacy Policy and Behaviour. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 14*(3), 253-266. https://doi.org/10.58346/JOWUA.2023.I3.019

[32] Thakor, V. A., Razzaque, M. A. & Khandaker, M. R. A. (2021). Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. *IEEE Access, 9*, 28177–28193. https://doi.org/10.1109/ACCESS.2021.3052867

[33] Ujjwal, R. L. & Singh, R. (2024). Intrusion Detection System based on Chaotic Opposition for IoT Network. *International Journal of Electrical and Computer Engineering Systems, 15*(2), 121–136. https://doi.org/10.32985/ijeces.15.2.1

[34] Wan, Q., & Hu, X. (2024). Legal Framework for Security of Organ Transplant Information in the Digital Age with Biotechnology. *Natural and Engineering Sciences, 9*(2), 73-93. https://doi.org/10.28978/nesciences.1569190

[35] Yang, W., Wang, R., Guan, Z., Wu, L., Du, X. & Guizani, M. (2020). A Lightweight Attribute Based Encryption Scheme with Constant Size Ciphertext for Internet of Things. *IEEE International Conference on Communications, 2020-June.* https://doi.org/10.1109/ICC40277.2020.9149294

Authors Biography

Yasamin Hamza Alagrash is a senior lecturer at Mustansiryah University, College of Science, Baghdad, Iraq. Holds ph. D degree in Computer Science and Informatics from Oakland University, USA, her areas of interest are cyber security, malware detection, machine learning, distributed system approaches, cyber threat intelligence.

Salah Taha Allawi, earned a B.Sc. in Computer Science from Mustansiriyah University in 1992 and an MSc in Computer Science from the Iraq Commission for Computers & Informatics, Informatics Institute for Postgraduate Studies in 2004. He is an associate professor at the Computer Science Department, College of Science, Mustansiriyah University. His research areas are image processing and information security. He has published many research papers as an author in local and international refereed journals.

Jamal Naser Hasoon received the PhD in Computer Science (University of Technology, Iraq) in 2020 and finished his Master and BSc degrees (Mustansiriyah University, Iraq) in 2014 and 2006, respectively. His research interests are machine learning, computer vision, pattern recognition, and information security.