future internet

MDPI

Article

Performance Evaluation of Lightweight Stream Ciphers for Real-Time Video Feed Encryption on ARM Processor

Mohsin Khan *, Håvard Dagenborg and Dag Johansen

Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway; havard.dagenborg@uit.no (H.D.); dag.johansen@uit.no (D.J.)
* Correspondence: mohsin.khan@uit.no

Abstract: In resource-intensive Internet of Things applications, Lightweight Stream Ciphers (LWSCs) play a vital role in influencing both the security and performance of the system. Numerous LWSCs have been proposed, each offering certain properties and trade-offs that carefully balance security and performance requirements. This paper presents a comprehensive evaluation of prominent LWSCs, with a focus on their performance and resource consumption, providing insights into efficiency, efficacy, and suitability in the real-world application of resource-intensive live video feed encryption on an ARM processor. The study involves the development of a benchmarking tool designed to evaluate key metrics, including encryption frame rate, throughput, processing cycles, memory footprint, ROM utilization, and energy consumption. In addition, we apply the E-Rank metric, which combines key performance and resource metrics to derive a unified comparative measure for overall software performance.

Keywords: lightweight cryptography; stream cipher; performance evaluation; Internet of Things; resource-intensive application

check for updates

Citation: Khan, M.; Dagenborg, H.; Johansen, D. Performance Evaluation of Lightweight Stream Ciphers for Real-Time Video Feed Encryption on ARM Processor. *Future Internet* **2024**, *16*, 261. https://doi.org/10.3390/fi16080261

Academic Editor: Xuebin Ren

Received: 31 May 2024
Revised: 13 July 2024
Accepted: 22 July 2024
Published: 25 July 2024

1. Introduction

The extensive adoption of Internet of Things (IoT) devices across industrial, scientific, commercial, and public sectors highlights their increasing importance in modern society. Their compact size and seamless integration capabilities enable a wide range of applications, leading to their rapid adoption. However, due to their limited capabilities, such as processing power, energy consumption, and memory resources, ensuring security for these devices is a significant challenge. While traditional cryptographic methods like RSA [1] and AES [2] offer security, their computational demands often make them impractical for resource-constrained IoT devices, leading to issues related to computational complexity, memory constraints, energy usage, and overall efficiency. To address these challenges, Lightweight Cryptography (LWC) emerged as a customized solution for resource-constrained devices [3].

LWC is categorized in two primary categories: Lightweight Block Ciphers (LWBCs) and Lightweight Stream Ciphers (LWSCs). LWBCs are designed for processing or encrypting discrete blocks, making them suitable for tasks like encrypting sensor data or enabling secure communication between IoT devices and edge gateways, while LWSCs are designed for handling continuous data streams, such as encrypting live video feeds or securing audio communication channels. Although LWSCs are specifically designed for encrypting continuous data streams, their suitability in resource-intensive IoT applications remains debatable, especially for devices with restricted memory, processing power, and energy consumption.

In this study, we aim to assess the performance of prominent LWSCs, including Grain-v1, Mickey, Trivium, Salsa, Sosemanuk, and an optimized version of Grain-128a. Through comprehensive performance analysis, we provide insights into the efficiency, efficacy, and suitability of these LWSCs for the real-time, resource-intensive encryption of live video feeds. The experiment was carried out using a Raspberry Pi Zero W, featuring a single-core ARM11 processor and enabling the seamless integration of the camera module. To precisely assess the performance and resource consumption of each selected LWSC, we developed a benchmarking framework that considers the following measurement metrics: encryption frame rate, throughput, Cycles per Byte (CpB), memory footprint, ROM utilization, and energy consumption. Energy consumption during LWSC operations is measured using an Arduino coupled with a power measurement sensor. Furthermore, we utilized the E-Rank metric established in our recently conducted study, derived from the Rank [4] metric and incorporating insights from the FoM [5] metric. This metric serves as a comprehensive and optimal means of evaluating overall software performance by balancing performance, energy consumption, and resource utilization into a unified comparative measure.

This paper is organized as follows: Section 2 offers a brief explanation of the structure of the selected LWSCs and reviews a few existing benchmarking tools, pointing out their limitations and explaining why they are not suitable for our research. Section 3 details our research methodology, including the development of the tool, selection criteria for LWSCs, hardware requirement factors, and benchmarking metrics. Sections 4 and 5 visually present our results through graphs and offer a detailed analysis and discussion on the implications and explanations behind the performance, resource usage, and overall software performance of the selected LWSCs. Finally, Section 6 presents a comprehensive summary of the main findings and insights from the study while also highlighting potential avenues for future research.

2. Background and Related Work

The European Network of Excellence for Cryptology (ECRYPT) Stream Cipher Project (eSTREAM) was launched in 2004 by the Information Society Technologies Programme of the European Commission with the aim of discovering new and more efficient stream ciphers for modern resource-restricted devices [6]. The program received submissions of 34 ciphers from various researchers, of which 7 were selected as finalists following a rigorous multi-tiered selection process. These submitted ciphers underwent assessment across three phases and were categorized into Profile 1 (software-oriented) and Profile 2 (hardware-oriented). Profile 1 comprises ciphers optimized for software applications requiring high throughput, while Profile 2 contains ciphers developed for hardware applications with constrained resources.

Profile 1 consists of four cryptographic algorithms: Rabbit [7], Salsa [8], HC-128 [9], and Sosemanuk [10]. Our research in this paper is aimed at conducting experiments specifically focused on the Salsa and Sosemanuk ciphers. Therefore, we will briefly introduce their structures. The Salsa cipher operates on 64-byte data blocks. It utilizes an expansion function that takes a secret key (128 or 256 bits) and combines it with a 64-bit nonce, followed by an additional block. The cipher employs a pseudorandom function based on a hash function core, utilizing operations such as addition modulo, bitwise XOR, and 32-bit constant distance rotation. The hash function processes the output obtained from the expansion function by operating on a sequence comprising the secret key, nonce with a block number, and four constant vectors from the expansion function. The resulting output is then XORed with the data block to produce the ciphertext. Salsa20 is built on a 20-round process and makes use of an invertible quarter-round function as its foundational component. Sosemanuk is a cryptographic algorithm that combines features from both synchronous stream ciphers and block ciphers. It offers flexible key length options, ranging from 128 to 256 bits, and utilizes a 128-bit initialization vector (IV). During the key setup phase, the algorithm loads the key into a 256-bit internal state and then expands it using the key scheduling algorithm, resulting in an intermediate key array. This array is used to initialize a Linear Feedback Shift Register (LFSR) and a Finite State Machine (FSM). Then, the LFSR and FSM, inspired by the design strategy of the SNOW stream cipher [11], leverage a 4-bit S-box inspired by the Serpent block cipher [12] to generate a keystream.

The LFSR enables rapid and efficient bit generation, while the FSM manages the non-linear transformation of the state. The resulting keystream is then used to XOR the plaintext, resulting in the ciphertext.

Profile 2 consists of three cryptographic algorithms: Trivium [13], Grain [14], and Mickey [15]. Trivium is designed with a focus on simplicity, utilizing only three shift registers and a few logical operations. The cipher operates in two distinct phases: initialization and keystream generation. In the initialization phase, the cipher loads both the 80-bit key and the 80-bit IV into an internal state of 288 bits. It then progresses through 1152 cycles to ensure thorough diffusion. During the keystream generation phase, the cipher generates output bits by applying a non-linear function that extracts specific bits from the internal registers and combines them using XOR and AND operations. These keystream bits are then XORed with plaintext bits to produce the ciphertext. Similarly, Grain-v1 follows a two-phase approach. In the first phase, the Non-linear Feedback Shift Register (NLFSR) is initialized with an 80-bit key, while the LFSR is set up with an 80-bit IV. During the second phase, the cipher generates each final keystream bit by combining feedback from both the NLFSR and LFSR through XOR operations with the corresponding input bit. Subsequently, each keystream bit is XORed with the corresponding plaintext bit to produce the ciphertext. Grain-128a improves upon the design of Grain-v1 by making significant enhancements to its security parameters. These include increasing the key size from 80 bits to 128 bits and the internal state from 160 bits to 256 bits, resulting in a higher level of cryptographic strength. Additionally, an authentication mechanism has been integrated to strengthen security and resilience against potential vulnerabilities further. The Mickey-v1 cipher is designed with a key size of 80 bits and an IV ranging from 0 to 80 bits for initializing the internal state, ensuring a higher degree of diffusion through initial mixing steps. The cipher enhances randomness by irregularly clocking shift registers. This irregular clocking mechanism, in combination with non-linear feedback from one register and linear feedback from another register, each with 80 stages, produces a complex and secure keystream. Each stage in these registers generates one bit of the keystream. The resultant keystream is then XORed with plaintext bits for encryption.

Our study commenced with a comprehensive assessment of existing benchmarking tools developed for evaluating the performance of LWSCs on resource-constrained devices. While some tools offered specific performance metrics, they exhibited constraints and operational deficiencies that were not aligned with our research objectives. Our research aims to evaluate the performance characteristics and resource utilization to examine the efficiency, efficacy, and suitability of LWSCs during the encryption of a resource-intensive task on resource-constrained IoT devices. Moreover, current tools face significant challenges due to the intricate network of platform interdependencies. This can lead to compatibility issues and hinder smooth operation across various systems. The complexity of the code adds to the difficulty of maintaining the tools, making it hard to manage and update them effectively. Also, there are significant constraints in integrating new ciphers, as these tools often lack the flexibility and adaptability required to accommodate other lightweight cryptographic algorithms.

The research study by Ertaul and Woodall [16] provides a detailed examination of the performance attributes of eSTREAM profile-II (hardware-oriented) ciphers, focusing on Grain-v1, Mickey, and Trivium. The evaluation includes important metrics such as throughput, memory utilization, and the power consumption of the ciphers. However, there are considerable limitations as the study does not cover eSTREAM profile-I ciphers (software-oriented) and does not provide insights into the performance of these ciphers under resource-intensive application scenarios.

In the study conducted by Deb and Bhuyan [17] and Gorbenko et al. [18], researchers undertake both statistical security evaluations and performance assessments of specific LWSCs. While statistical security evaluations provide valuable insights by analyzing the randomness of the generated keystream, performance evaluations offer information regarding throughput and memory consumption in the former case and only throughput in the latter case. However, neither study addresses energy consumption or the feasibility of using these LWSCs in resource-intensive applications, creating a gap in understanding their applicability.

The existing literature lays the groundwork for understanding the performance and characteristics of various LWSCs. However, current frameworks only focus on specific metrics and do not provide comprehensive assessments of the suitability of LWSCs for data-intensive tasks involving continuous data streams. There is an evident need for comprehensive benchmarking tools and methodologies to evaluate these ciphers. Our framework seeks to address these challenges by offering a customized solution in a controlled environment for evaluating LWSCs in a real-world, real-time, data-intensive scenario within a constrained IoT environment.

3. Research Methodology

In this experimental study, we present a structured methodology for assessing the software performance of LWSCs while encrypting resource-intensive live video feeds on an ARM11 processor. The research methodology provides the development of a benchmarking tool specifically customized for ARM11 processors, which involves a thorough analysis of requirements, the establishment of implementation criteria, rigorous testing, debugging, and optimization. LWSCs are carefully selected based on their capability to address IoT security challenges and their importance in the lightweight cryptographic community. Additionally, specific implementation considerations for ARM processors are outlined to ensure limitations in computational power and memory capacity for proving a resource-constrained IoT environment. Evaluation metrics, including encryption frame rate, throughput, processor cycles, memory footprint, flash memory usage, and energy consumption, are defined and scrutinized to comprehensively assess the efficacy, efficiency, and suitability of LWSCs in resource-intensive applications.

3.1. Development of Benchmarking Tool

The benchmarking tool was designed to be lightweight, adaptable, and easy to use. We utilized Python as the foundation and integrated the C implementation of ciphers via shared object files, thereby enhancing its capabilities. By integrating shared object files with Python, we established a straightforward and efficient solution for cryptographic operations and system performance monitoring. In the upcoming sections, we will provide an elaborate overview of the development process, covering each stage and the methodologies implemented.

3.1.1. Requirement Analysis

The initial step focused on a comprehensive assessment of the necessary requirements and goals essential for developing the benchmarking tool. In this phase, we determined the specific characteristics, input parameters, and output evaluation metrics needed for the benchmarking tool. Based on these requirements, we created a detailed design for the benchmarking tool, which is illustrated in Figure 1. This design shows the structure and workflow of the tool, including modules for input processing, basic cryptographic functions, and key evaluation metrics.

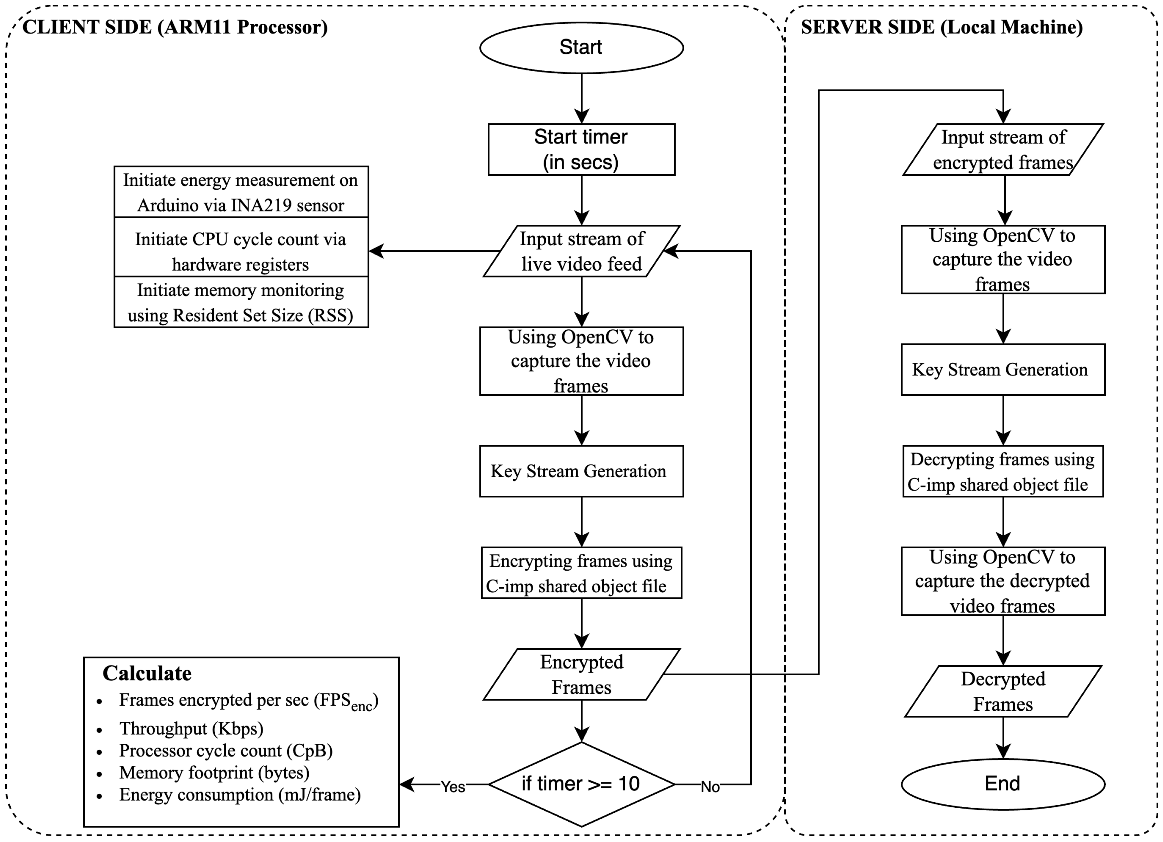

CLIENT SIDE (ARM11 Processor)
Start
Start timer (in secs)
Initiate energy measurement on Arduino via INA219 sensor
Initiate CPU cycle count via hardware registers
Initiate memory monitoring using Resident Set Size (RSS)
Input stream of live video feed
Using OpenCV to capture the video frames
Key Stream Generation
Encrypting frames using C-imp shared object file
Encrypted Frames
if timer >= 10
Yes
No
Calculate
Frames encrypted per sec (FPS_{enc})
Throughput (Kbps)
Processor cycle count (CpB)
Memory footprint (bytes)
Energy consumption (mJ/frame)
SERVER SIDE (Local Machine)
Input stream of encrypted frames
Using OpenCV to capture the video frames
Key Stream Generation
Decrypting frames using C-imp shared object file
Using OpenCV to capture the decrypted video frames
Decrypted Frames
End

Figure 1. Flowchart illustrating the encryption and decryption processes between the client and server, along with the initialization of performance tracking on the ARM11 processor (client side).

3.1.2. Implementation

During the implementation phase, our primary objective was to translate the design and architecture of the benchmarking tool into operational code. This involved integrating the C implementations of the selected LWSCs with Python utilizing the *ctypes* library [19]. By doing so, we enabled the direct invocation of C-implemented shared object functions from Python. This approach was designed to maximize the speed and efficiency of the tool by harnessing the swift processing power of C language for cryptographic operations while leveraging Python's extensive capabilities for orchestrating system procedures and handling data to capture key evaluation metrics. The decision to utilize Python as the programming platform for performance measurements was driven by its favorable attributes, such as simplicity, a rich library ecosystem, and rapid prototyping capabilities. During this phase, several key functionalities were implemented, including the following:

- Input Processing: The system is designed to prompt user input specifying the selected LWSC at the server side and client side. The client-side comprises the Raspberry Pi Zero connected with a camera module, while the server side consists of the local machine connected through the local network. The user input prompt invokes the OpenCV (Open Source Computer Vision Library) [20], which provides a wide range of functionalities for image and video processing tasks. In our case, we opted for the stripped-down version of the OpenCV library on the client side to accommodate the limited resources of the Raspberry Pi Zero. The program uses the library to capture frames for input processing (encryption) on the client side. Subsequently, these encrypted frames are transmitted to the server side, which may function as an edge node; however, in our case, we are conducting the experiment within a local network environment. On the server side, the received encrypted frames serve as input for decryption processing.

- Cryptographic Operations: The process of encrypting and decrypting live video feed involves carrying out specific operations while using defined parameters. This is done by incorporating the shared object file that is created from the C implementation of the selected LWSCs.
- Performance Measurement: The benchmarking tool has been specifically designed to integrate libraries, functions, and system calls for the purpose of measuring a wide range of key performance and resource consumption metrics. These metrics encompass encryption frame rate, throughput, CpB, memory usage, and energy consumption. By collecting data on these metrics, the tool facilitates a comprehensive comparison of selected LWSCs. This quantitative comparison provides a comprehensive evaluation of cipher performance in resource-intensive applications, specifically in live video feed encryption under controlled experimental conditions.
- Result Analysis: The data obtained from the benchmarking tool are used to create graphical illustrations that showcase the performance of each LWSC across various evaluation metrics. To ensure the accuracy of these graphs, we establish the confidence interval by evaluating the performance of each cipher iteratively over multiple distinct time intervals, using a 95% confidence level. This statistical technique provides a deeper understanding of performance consistency and potential variability, ultimately enhancing the interpretive value of the data. Incorporating confidence intervals into the bar graphs enables us to gain detailed insights into the stability and consistency of each LWSC.

3.1.3. Testing and Debugging

During this phase, we conducted multiple tests to enhance the stability and reliability of the benchmarking tool. Regression testing was performed to ensure that new changes did not disrupt the existing functionality of the tool. We also performed statistical analysis, as explained earlier, to validate the accuracy of the data, which involved checking for consistency and stability in data outputs. Lastly, load testing allowed us to evaluate the system's performance under various stress conditions, including scenarios such as increasing the frame rate or implementing image filtration using the OpenCV library. Through comprehensive validation against established benchmarks and reference implementations, we ensured the accuracy of the tool. The feedback we received during this phase was essential in identifying and addressing any issues, bugs, or inconsistencies, resulting in a significant enhancement in the overall effectiveness of the tool.

3.1.4. Optimization and Performance Tuning

We improved our benchmarking tool by implementing optimization techniques to enhance efficiency and effectiveness. One key aspect of this involved integrating efficient and validated C implementations of LWSCs, which we acquired from their respective research implementations. This allowed us to carry out essential cryptographic operations at faster speeds while ensuring low-level access. Similarly, we approached the selection and use of necessary libraries carefully to improve performance measurement accuracy and minimize resource usage. As a result, we were able to effectively reduce unnecessary overhead and significantly improve the responsiveness of the tool without increasing its overall resource demands. Utilizing these optimization techniques is essential for evaluating performance in resource-constrained IoT environments and ensuring the reliable transfer of encrypted data streams.

3.2. Selection Criteria for Lightweight Stream Ciphers

In this study, we selected six LWSCs for evaluation based on their relevance to the eSTREAM project. The selection includes Grain-v1, Trivium, Mickey, Salsa, and Sosemanuk, all of which were finalists in the ECRYPT Stream Cipher Project, demonstrating their robust cryptographic strength and operational efficiency. We also integrated an optimized version of Grain-128a, which includes speed enhancements and strengthened security features. This diverse set of LWSCs enables a comprehensive examination of cryptographic methodologies suitable for resource-intensive applications on resource-constrained IoT devices, focusing on their performance characteristics and resource utilization.

3.3. Implementation Considerations for ARM-Processor

The benchmarking tool has been optimized for deployment on the Raspberry Pi Zero, which is characterized by its single-core ARM11 processor [21] and seamless integration with the camera module. This processor was selected due to its limitations in computational power and memory capacity, which are key factors in simulating the performance characteristics of lightweight cryptographic algorithms in constrained environments. The Raspberry Pi Zero is equipped with 512 MB of RAM and effectively manages memory to thwart bottlenecks during the processing of resource-intensive live video stream feed. The header on the device is strategically utilized, with specified pins used for providing power to the device and for serial data transfer. This enables the use of the transfer (Tx) line on the Raspberry Pi to communicate with the Arduino. This is essential for indicating the start and end of power and energy measurements at the beginning and end of each cryptographic process. This configuration ensures the precise tracking of the Raspberry Pi's energy consumption while executing the selected LWSC. Further explanation of this connection and its role in the benchmarking framework will be provided in the subsequent section on evaluation metrics.

3.4. Evaluation Metrics

The following metrics were used in our evaluation: encryption frame rate, throughput, CpB, code size, memory footprint, energy consumption, and E-Rank.

Encryption Frames per Second ($\mathbf{FPS_{enc}}$) measures the number of frames that can be encrypted within a second using selected LWSCs. FPS_{enc} presents a mechanism to compare the efficacy of LWSCs and assess their suitability for encrypting real-time, resource-intensive live video feeds. This metric is essential for determining which ciphers can meet the resource-intensive requirements of live video feed encryption on resource-constrained IoT devices. Higher values of FPS_{enc} indicate a greater number of frames being encrypted per second, indicating better efficacy and suitability.

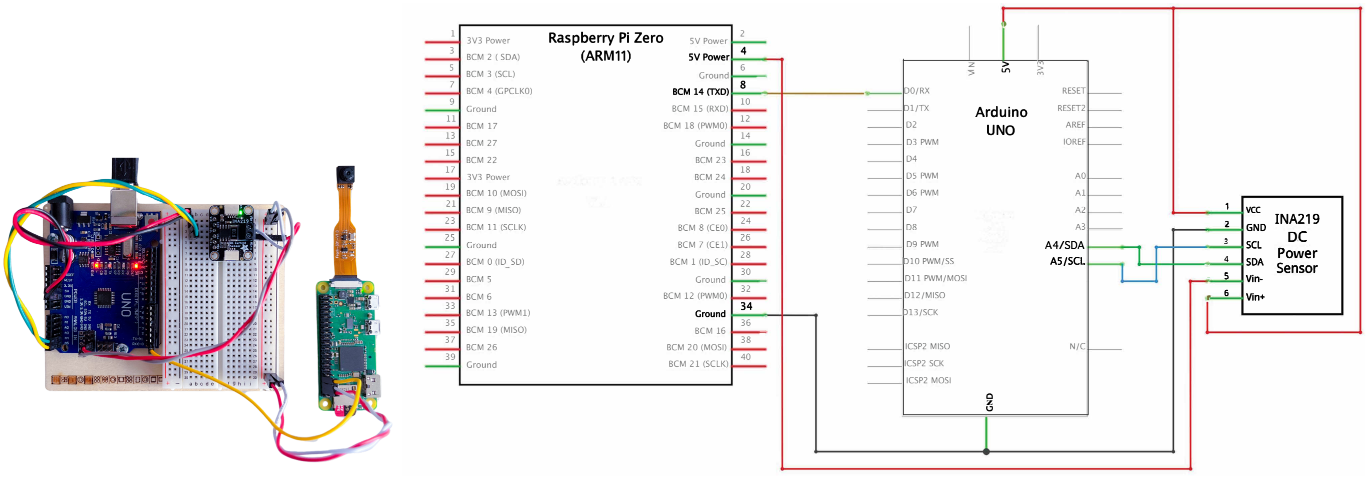
Throughput is defined as the rate at which data are processed within a given period of time. In the context of this paper, throughput is defined as the speed at which an LWSC can encrypt a live video feed, measured in kilobits per second (Kbps). In resource-intensive applications like live video streaming, real-time surveillance, or high-speed data transfers, a higher throughput value represents a faster rate of encryption or decryption, which is important for maintaining optimal system performance and responsiveness.

Cycles per Byte (CpB) represents the average count of processing cycles needed to handle each byte of data. To enhance the precision of CpB measurements for LWSCs on the ARM11 processor, we implemented a specialized approach. This approach involves direct access to the ARM timer registers and system timer registers, which are integral for managing various timing and scheduling operations within the Raspberry Pi's Linux-based operating system. Our program directly interfaces with hardware registers to capture timer values, enabling the precise measurement of processor cycles. The system calculates a baseline CpB exclusively for background processes running on the processor and averages it over time. During LWSC execution, CpB is derived for that specific cipher by subtracting this baseline CpB from observed CpB values at regular intervals, ensuring accurate measurement and mitigating the processor cycles utilized by the background processes. Lower CpB values in the context of LWSCs indicate faster performance, as they signify that fewer processing cycles are required for the encryption or decryption of a continuous stream of data.

Code Size represents the volume of memory consumed in the non-volatile storage. In the context of the Raspberry Pi Zero, it specifies the storage capacity occupied by each LWSCs in the flash memory of the device, expressed in bytes.

Memory Footprint refers to the memory utilized by a process throughout its operation. In our methodology, the memory footprint of LWSCs is calculated by evaluating the volatile memory utilized by each cipher and dividing it by the size of the input data stream of the video feed, thereby quantifying the amount of volatile memory (measured in bytes) required to process each byte of input. This process leverages the Resident Set Size (RSS), which specifies the memory allocated to a process in the main memory. Within this research framework, RSS represents the allocation of memory by LWSCs in primary storage, encompassing critical segments like the code segment, data segment, heap, and stack. This approach provides a holistic view of actual memory usage by each LWSC, excluding any data stored on disk. The memory footprint provides valuable insights into the actual memory requirements of the ciphers and how they impact the allocation of system resources.

Energy Consumption refers to the power consumption of a process during its operation over a defined duration of time. In the context of this study, it quantifies the energy consumption of selected LWSCs on the ARM11 processor, measured in millijoules per frame (mJ/frame). This metric enables the assessment of the energy consumption of individual frames, providing insights into the energy efficiency of the encryption process at a granular level. The diagram illustrated in Figure 2b showcases a circuit schematic that has been specifically designed for the purpose of measuring power and calculating energy. Power is supplied to the Raspberry Pi board by an Arduino UNO through GPIO pin-4 (5V) and pin-6 (ground). The power line connection between the Arduino and the Raspberry Pi incorporates the INA219 sensor from Texas Instruments [22], which accurately monitors power consumption in real time, and Arduino facilitates the computation and logging of energy consumption. Furthermore, one-way serial communication is established by connecting GPIO pin-14 on the Raspberry Pi to pin-0 on the Arduino. This communication channel serves to initiate and terminate measurements related to power and energy consumption. At the beginning of the encryption process, the Raspberry Pi sends the name of the cipher through the serial communication channel to signal the Arduino to activate the INA219 to start recording power consumption. After the encryption process is completed, another message is sent to the Arduino to signal the termination of power and energy consumption measurement for that specific LWSC. In order to accurately calculate the energy consumption of a specific LWSC while excluding the influence of background processes, the power consumption of the Raspberry Pi is first measured during an initialization phase where only background processes are active. This average power consumption is then subtracted from the current readings while the cipher is executed at discreet time intervals, isolating the power utilized exclusively by that specific LWSC.



(**a**) Physical configuration (**b**) Schematic diagram of the connection setup

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

Rank provides a unified metric for software performance by balancing performance and resource consumption in lightweight cryptographic ciphers. The term was initially introduced by Beaulieu et al. [4] and refers to the ratio between the speed at which data are processed and the amount of memory, including both volatile and non-volatile memory, utilized during the encryption process, as illustrated in Equation (1). However, in the context of this research study, we utilized a modified version of Rank, denoted as E-Rank, which incorporates insights from the Figure of Merit (FoM). The FoM was introduced by Badel et al. [5] to evaluate their LWBC ARMADILLO, and it typically assesses hardware performance, but the equation does not account for power dissipation, as indicated in Equation (2). The authors clarify the exclusion of power dissipation by explaining the direct relationship between dynamic power and total switched capacitance. In hardware implementation of cryptographic ciphers, power is directly linked to the number of gates, as each transistor within a gate contributes to the overall capacitance. When the number of gates increases in a circuit, the total switched capacitance also increases proportionally. The calculation of the Rank metric depends on total memory consumption, which does not have a direct correlation with switched capacitance. Therefore, the incorporation of energy measurement into the Rank provides a more refined evaluation metric for overall software performance. This refined metric is called E-Rank and is presented in Equation (3). During the calculation of E-Rank, applying unit normalization ensures the consistent expression of all metrics in bytes.

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

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

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

4. Experimental Results

4.1. Frames Encrypted per Sec (FPS$_{enc}$) Analysis

Figure 3 presents the real-time video feed frames encrypted per second (FPS$_{\text{enc}}$) by the selected LWSCs. The results demonstrate that Salsa and Sosemanuk outperform the other ciphers, achieving the highest frame encryption rates. Grain-128a exhibits moderate performance. In contrast, Grain-v1, Mickey-v1, and Trivium have the lowest FPS$_{\text{enc}}$, increasing in the aforementioned order.

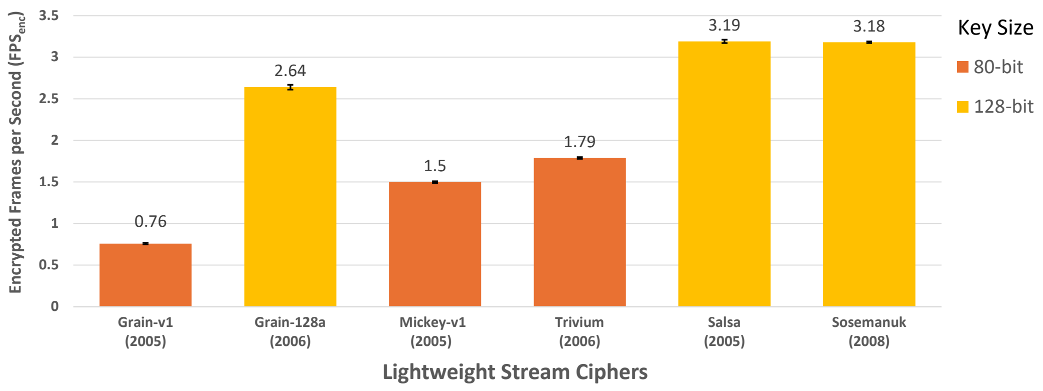


Figure 3. Frames encrypted per second by the selected Lightweight Stream Ciphers (LWSCs).

4.2. Throughput Analysis

Figure 4 illustrates the throughput achieved by the selected LWSCs while encrypting the video feed stream. Sosemanuk has the highest throughput, followed by Salsa and Grain-128a. In contrast, Grain-v1, Mickey-v1, and Trivium have the lowest throughputs, with their performance improving in the specified order.

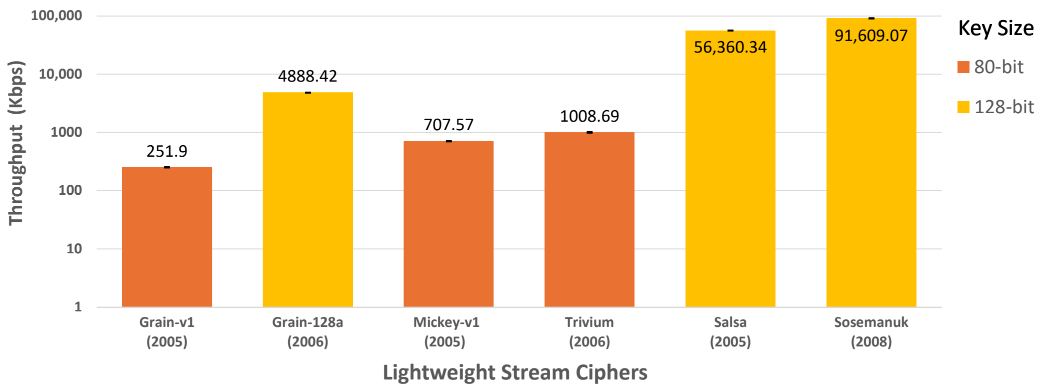


Figure 4. Throughput of selected LWSCs.

4.3. CpB Analysis

Figure 5 shows the number of processor cycles required per byte for the encryption of the real-time video feed by the selected LWSCs. Grain-v1 exhibits the highest number of CpB, indicating relatively higher computational demands. Trivium and Mickey follow with notable differences in processor cycle consumption compared to Grain-v1. In contrast, Sosemanuk has the lowest CpB, whereas Salsa and Grain-128a demonstrate slightly higher cycle counts compared to Sosemanuk, without any significant differences.

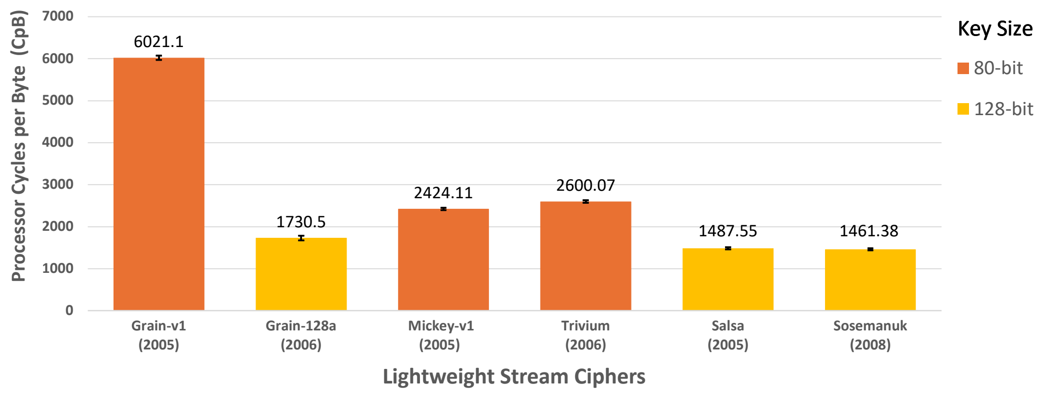


Figure 5. Processor Cycles per Byte (CpB) of selected LWSCs.

4.4. Memory Consumption Analysis

In Figure 6, Salsa demonstrates the largest memory footprint, followed by Grain-128a, while Sosemanuk exhibits the smallest footprint, followed by Grain-v1. Mickey-v1 and Trivium have relatively moderate memory footprints.

In Figure 7, Sosemanuk displays the highest code size, significantly differing from the other selected LWSCs, with Grain-128a following behind. Grain-v1 has the smallest code size, followed by Salsa, Trivium, and Mickey.

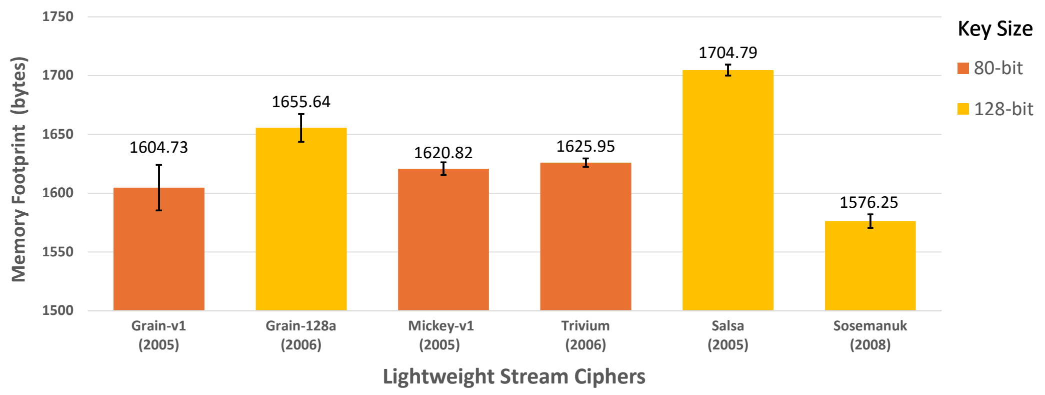


Figure 6. Memory footprint of selected LWSCs.

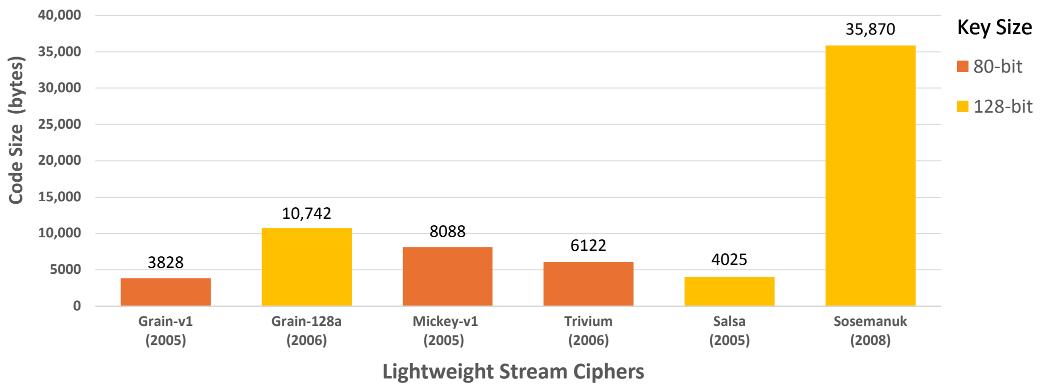


Figure 7. Code size of selected LWSCs.

4.5. Energy Consumption Analysis

Figure 8 presents the energy consumption measurements of selected LWSCs during the encryption of individual frames within live video feeds. Grain-v1 exhibits the highest energy requirement per frame, followed by Mickey and Trivium. In contrast, Sosemanuk, Salsa, and Grain-128a demonstrate relatively lower energy requirements for frame encryption, with subtle distinctions observed among them.

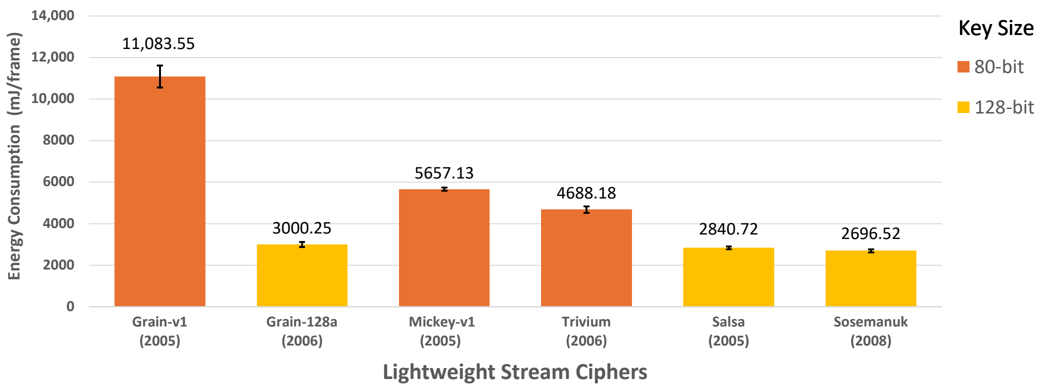


Figure 8. Energy consumed per frame by the selected LWSCs.

4.6. E-Rank Analysis

In Figure 9, the graph illustrates the E-Rank values for the selected LWSCs. Salsa exhibits the highest E-Rank, with a notable lead over Sosemanuk. Contrarily, Grain-v1 displays the lowest E-Rank, with Mickey and Trivium following behind. Grain-128a shows a relatively moderate E-Rank, positioning it in the middle tier of overall software performance among the selected LWSCs.

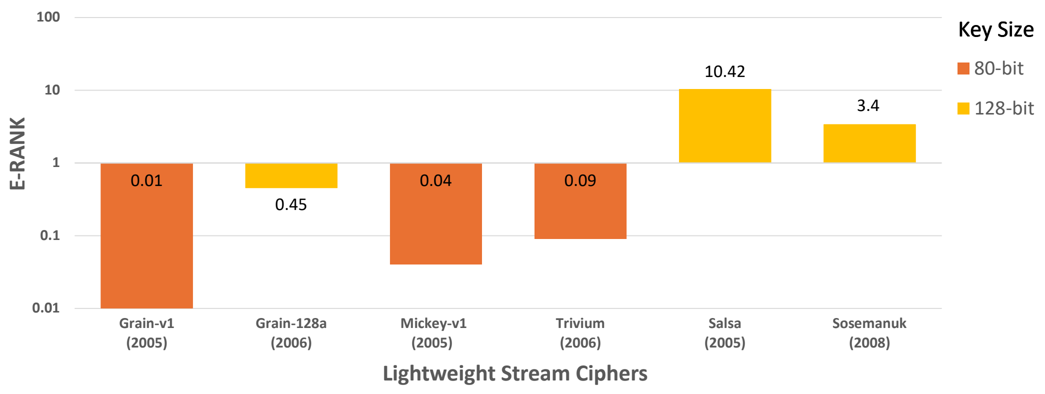


Figure 9. E-Rank of selected LWSCs.

5. Discussion

This study methodically evaluated the performance and resource usage of LWSCs using a comprehensive benchmarking framework optimized for a resource-intensive application, i.e., live video feed encryption on a resource-constrained ARM11-based IoT device. Our analysis encompassed all key metrics, including frames encrypted per second (FPS_{enc}), processor cycles, energy consumption, and total memory utilization. The latter includes both the memory footprint and code size. These metrics collectively provide detailed insights into the performance and resource demands of the LWSCs, offering a comprehensive understanding of their efficacy, efficiency, and suitability for real-time, resource-intensive applications.

The performance analysis of the selected LWSCs, as illustrated in Table 1, indicates that Sosemanuk and Salsa demonstrate the highest throughput, lowest CpB, and lowest energy consumption. This can be attributed to their software-optimized architectures. Sosemanuk incorporates features of both synchronous stream and block ciphers, allowing the more direct mapping of data on the processor registers, which improves its initialization speed and performance efficiency. Salsa, on the other hand, utilizes a refined structure involving addition modulo, bitwise XOR, and constant distance rotation, making it highly efficient for software implementation. This demonstrates that Salsa and Sosemanuk leverage arithmetic and logical operations that align effectively with CPU instructions, enabling them to execute efficiently within software implementation. Contrarily, hardware-oriented LWSCs such as Grain, Mickey-v1, and Trivium generally exhibit lower software performance efficiency. The efficiency of these structures arises from their utilization of shift registers and feedback loops, which can be implemented using a minimal number of transistors that can operate in parallel with high efficiency in hardware. However, adapting this parallel operation to a sequential execution model in software leads to inefficiencies in ARM11-based processors. Moreover, these ciphers function at the bit level, which poses challenges for ARM-based general-purpose CPUs that are designed to handle operations at the byte level. Handling individual bits in software can lead to more intricate and time-consuming operations. For example, Grain, Trivium, and Mickey each undergo two phases: initialization and keystream generation. During initialization, these ciphers load the key and initialization vector into their internal state. Grain and Mickey utilize two shift registers for their operations, while Trivium employs three shift registers. These shift registers, in combination with feedback and non-linear functions, are responsible for generating the keystream. However, when these ciphers are implemented on ARM11-based processors, the need for complex bit manipulation operations like shifting and applying non-linear functions can lead to decreased cryptographic throughput, increased processing cycles, and energy consumption. This is particularly evident in ARM-based byte-oriented CPUs. An exception in this experimentation is Grain-128a, which outperforms the selected hardware-oriented LWSCs due to its speed-optimized design.

Table 1. Software implementation results of Lightweight Stream Ciphers (LWSCs) performed on ARM processor along with E-RANK.

Ciphers	Key Size (Bits)	Frames Enc per Sec (FPS_{enc})	Code Size (Bytes)	Memory Footprint (Bytes)	Processor Cycles (CpB)	Throughput (Kbps)	Energy (mJ/Frame)	Energy (μJ/B)	E-RANK
Grain-v1	80	0.76	3828	1604.73	6021.1	251.9	11,083.55	354.98	0.01
Grain-128a	128	2.64	10,742	1655.64	1730.5	4888.42	3000.25	96.09	0.45
Mickey-v1	80	1.5	8088	1620.82	2424.11	707.57	5657.13	181.18	0.04
Trivium	80	1.79	6122	1625.95	2600.07	1008.69	4688.18	150.15	0.09
Salsa	128	3.19	4025	1704.79	1487.55	56,360.34	2840.72	90.98	10.42
Sosemanuk	128	3.18	35,870	1576.25	1461.38	91,609.07	2696.52	86.36	3.4

The detailed analysis of resource consumption reveals that Sosemanuk exhibits the highest code size, with Grain-128a and Mickey-v1 following closely behind. In terms of memory footprint, Salsa has the highest footprint, followed by Grain-128a in second place. This thorough analysis underscores a recurring trend wherein lightweight ciphers that have the leading efficiency in performance, including throughput, CpB, and energy, demand a greater allocation of memory resources. Consequently, there is a clear trade-off that requires careful lightweight cipher selection to achieve a balanced performance in throughput, memory usage, and energy consumption. This ensures the efficient operation of resource-intensive applications on resource-constrained IoT devices. This can be delivered through the E-Rank metric that enables the effective integration of performance metrics with resource utilization, offering a comprehensive evaluation of the overall software performance of LWSCs. According to the E-Rank, Salsa demonstrates an optimal performance in terms of throughput, memory usage, and energy consumption, followed by Sosemanuk and then Grain-128a. These LWSCs effectively optimize resource consumption and uphold high-performance standards, making them favorable for resource-intensive IoT applications and, in the context of this paper, live video feed encryption. Grain-v1, Mickey-v1, and Trivium showed the lowest E-Rank, and thus, while using those ciphers in the resource-intensive live video feed encryption, these LWSCs demonstrated processing bottlenecks, scalability issues, and security risks due to prolonged encryption spans. The speed-optimized version of Grain-128a demonstrated a higher E-Rank compared to Grain-v1, Mickey, and Trivium. Therefore, it displayed intermediate favorability for encrypting resource-intensive live video feeds on resource-constrained ARM11 processors.

The efficacy and suitability of ciphers are observed through the FPS_{enc} metric. This analysis reveals that Salsa, Sosemanuk, and the optimized version of Grain-128a are notably more suitable and effective for resource-intensive applications, as demonstrated by their encryption frame rates. This characteristic enables reduced latency during the transmission of continuous encrypted data streams from resource-constrained devices to other devices within a network.

6. Conclusions and Future Work

In this paper, we introduced an optimized benchmarking framework designed for ARM processors to assess the efficiency, efficacy, and suitability of selected LWSCs for resource-intensive IoT applications. Sosemanuk, followed by Salsa, demonstrated the lowest CpB, highest throughput, and lowest energy consumption, indicating the best performance efficiency among the selected LWSCs. In addition, Sosemanuk and Grain-v1 have the smallest memory footprint, while Grain-v1 and Salsa exhibit the smallest code size. This indicates that these ciphers are efficient in terms of resource consumption. These results emphasize the trade-offs involved between efficiency in performance and resource consumption that are inherent in lightweight cryptographic solutions designed for resource-constrained environments. The E-Rank metric offered valuable insights by effectively balancing performance measurement with resource utilization, demonstrating Salsa and Sosemanuk's superior overall software performance due to its optimal and efficient balance between performance and resource consumption. The analysis conducted through the FPS_{enc} metric revealed that Salsa, Sosemanuk, and the speed-optimized version of Grain-128a exhibit better suitability and efficacy for the real-time encryption of continuous stream of live video feed on resource-constrained IoT devices.

The experimental setup used in this study was customized for specific hardware and software environments to allow for the precise measurement and analysis of LWSCs for resource-intensive applications in resource-constrained IoT environments. While it may not fully capture the complexities and diverse conditions in real-world IoT deployments, we recognize the importance of considering challenges such as network variability, environmental factors, and diverse hardware configurations of IoT devices. In future work, we aim to address these limitations by validating our findings in more realistic settings, including field deployment in actual IoT networks, simulating environmental conditions, and conducting long-term continuous evaluations.

Author Contributions: Conceptualization, M.K., H.D. and D.J.; methodology, M.K., H.D. and D.J.; software, M.K.; validation, M.K.; formal analysis, M.K.; investigation, M.K.; resources, H.D. and D.J.; data curation, M.K.; writing—original draft preparation, M.K.; writing—review and editing, H.D. and D.J; visualization, M.K.; supervision, H.D. and D.J.; project administration, H.D.; funding acquisition, H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Diffie, W.; Hellman, M.E. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*; ACM: New York, NY, USA, 2022; pp. 365–390.
2. Daemen, J.; Rijmen, V. *AES Proposal: Rijndael*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 1999. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf (accessed on 21 July 2024).
3. Dhanda, S.S.; Singh, B.; Jindal, P. Lightweight cryptography: A solution to secure IoT. *Wirel. Pers. Commun.* **2020**, *112*, 1947–1980. [CrossRef]
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 Proceedings of the Lightweight Cryptography for Security and Privacy: Third International Workshop, LightSec 2014, Istanbul, Turkey, 1–2 September 2014; Revised Selected Papers 3; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–20.
5. Badel, S.; Dağtekin, N.; Nakahara, J., Jr.; Ouafi, K.; Reffé, N.; Sepehrdad, P.; Sušil, P.; Vaudenay, S. ARMADILLO: A multi-purpose cryptographic primitive dedicated to hardware. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–20 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 398–412.
6. Robshaw, M. The eSTREAM project. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1–6.
7. Boesgaard, M.; Vesterager, M.; Pedersen, T.; Christiansen, J.; Scavenius, O. Rabbit: A new high-performance stream cipher. In Proceedings of the Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, 24–26 February 2003; Revised Papers 10; Springer: Berlin/Heidelberg, Germany, 2003; pp. 307–329.
8. Bernstein, D.J. The Salsa20 family of stream ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 84–97.
9. Wu, H. The stream cipher HC-128. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 39–47.
10. Berbain, C.; Billet, O.; Canteaut, A.; Courtois, N.; Gilbert, H.; Goubin, L.; Gouget, A.; Granboulan, L.; Lauradoux, C.; Minier, M.; et al. Sosemanuk, a fast software-oriented stream cipher. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 98–118.
11. Ekdahl, P.; Johansson, T. SNOW-a new stream cipher. In Proceedings of the First Open NESSIE Workshop, KU-Leuven, Leuven, Belgium, 13–14 November 2000; pp. 167–168.
12. Biham, E.; Anderson, R.; Knudsen, L. Serpent: A new block cipher proposal. In Proceedings of the International Workshop on Fast Software Encryption, Paris, France, 23–25 March 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 222–238.

13. De Canniere, C.; Preneel, B. Trivium. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 244–266.
14. Hell, M.; Johansson, T.; Meier, W. Grain: A stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.* **2007**, *2*, 86–93. [CrossRef]
15. Babbage, S.; Dodd, M. The MICKEY stream ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 191–209.
16. Ertaul, L.; Woodall, A. IoT security: Performance evaluation of grain, mickey, and trivium-lightweight stream ciphers. In Proceedings of the International Conference on Security and Management (SAM), Las Vegas, NV, USA, 17–20 July 2017; pp. 32–38.
17. Deb, S.; Bhuyan, B. Performance analysis of current lightweight stream ciphers for constrained environments. *Sādhanā* **2020**, *45*, 256 . [CrossRef]
18. Gorbenko, I.; Kuznetsov, A.; Gorbenko, Y.; Vdovenko, S.; Tymchenko, V.; Lutsenko, M. Studies on statistical analysis and performance evaluation for some stream ciphers. *Int. J. Comput.* **2019**, *18*, 82–88. [CrossRef]
19. Python Software Foundation. *Python Ctype Library Documentation*; Python Software Foundation: Amsterdam, The Netherlands, 2024.
20. Itseez. *The OpenCV Reference Manual*, 2.3.1 ed.; OpenCV.org: Willow Garage, CA, USA, 2011. Available online: https://www.opencv.org.cn/opencvdoc/2.3.1/opencv2refman.pdf (accessed on 21 July 2024).
21. ARM. *ARM1176JZF-S Technical Reference Manual r0p7*; ARM Limited: Cambridge, UK, 2009.
22. Texas Instruments. INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I2C Interface, 2021. Available online: https://www.ti.com/lit/ds/symlink/ina219.pdf (accessed on 21 July 2024).