



DESIGN AND EVALUATION OF HSPECK-WIDEPIPE: A NEW LIGHTWEIGHT CRYPTOGRAPHIC HASH ALGORITHM FOR RESOURCE-CONSTRAINED ENVIRONMENTS

Roman Alex F. Lustro*¹ and Daniel A. Neri²

¹College of Engineering, Architecture, and Technology, Isabela State University, City of Ilagan, Isabela, Philippines.

²College of Information System, Don Mariano Marcos Memorial State University, Bacnotan, La Union, Philippines.

¹<https://orcid.org/0000-0002-9350-5117>, ²<https://orcid.org/0000-0002-5588-3096>

Email: *romanalex.f.lustro@isu.edu.ph, dneri@dmmsu.edu.ph

ARTICLE INFO

Article History

Received: March 31, 2026

Reviewed: August 3, 2026

Accepted: September 3, 2026

Published: September 30, 2026

Keywords:

WidePipe construction,
ARX-based permutation,
microarchitectural profiling,
lightweight cryptography
Statistical Randomness Analysis

ABSTRACT

This study introduced hSpeck-WidePipe, a new, fast, and lightweight hash function. It combines ARX-based permutation and widepipe construction employing expanded internal state to strengthen security margins. Implemented across resource-constrained platforms, including Arduino Uno, Arduino Mega, and ESP8266, with RFID-based input integration. The evaluation framework combines microarchitectural profiling, hardware benchmarking, statistical analysis of randomness validation, and time and space complexity. Experiments prove microarchitectural profiling efficiency where hSpeck-WidePipe 128 achieved ≈ 0.78 instruction per cycle count. The 192 and 256 variants tallied 0.82 IPC which are slightly faster at more consistent. The new algorithm delivers up to 2–6 $\times$ speed improvement over conventional hash functions on Arduino Uno and 7–9 $\times$ faster execution on ESP8266. Similarly, it reduces energy consumption by up to 6–9 $\times$. The proposed method produces statistically uniform outputs across Dieharder evaluations and exceeds 98.8% in NIST statistical tests. Additionally, confirms scalability recording low memory overhead ~ 3 –4 KB exhibiting near linear time and space complexity. Overall, this work contributes a novel integration of ARX-based encryption within a Wide-Pipe hashing framework. Offering a balance between security and efficiency. The findings demonstrate suitability for secure IoT, embedded systems, and lightweight blockchain applications.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

Billions of interconnected devices operate within distributed systems due to the booming Internet of Things. These devices are used in various applications, including smart cities, industrial automation, intelligent systems, healthcare monitoring, and environmental sensing. However, this integration introduces cybersecurity challenges. The IoT devices operate with constrained memory, energy, and computational capabilities. Thus, limits the probability of employing a traditional security algorithm. Ensuring data integrity through secure communication has become a critical research topic in today's cybersecurity and embedded systems [1], [2]. Providing confidentiality, integrity, authentication, and non-repudiation in embedded systems means protecting information. These make hash functions indispensable [3], [4]. Various conventional cryptographic hash functions have been advanced to meet the growing needs of secure communication. SHA-2, SHA-3, and BLAKE are widely deployed and adopted hash functions that provide strong cryptographic security and have undergone extensive cryptanalytic scrutiny. However, these hash functions demand computational overhead, high memory consumption, and significant energy requirements, leading to unacceptable latency, power consumption, or resource exhaustion [5]. The field of lightweight cryptography has emerged as a research area addressing security optimization for resource-constrained devices. The lightweight cryptographic algorithms aim to limit application intricacy while preserving security and performance efficiency. Several lightweight block ciphers have been proposed, such as PRESENT, SIMON, and SPECK, as well as hash functions including PHOTON, SPONGENT, LWH-128 [6].

These designs enable practical deployment while attempting to balance security, efficiency, and implementation in a resource-constrained environment. The National Institute of Standards and Technology (NIST) acknowledges and strengthens the significance of lightweight cryptography. This underscored the need for an efficient cryptographic algorithm for protecting embedded systems and IoT infrastructure [7]. Despite these developments, recent studies still accentuate the limited investigation on lightweight block ciphers and hash functions [8]. The context of lightweight cryptography relies on simple ARX-based operations such as modular addition, bitwise rotation, and exclusive OR. These operations allow computational efficiency and while tallying low implementation overhead. One notable example is the Speck family of lightweight block cipher enabling performance efficiency and minimal code footprint when deployed on battery operated devices [9]. The efficiency advantages of ARX-based encryption algorithms remain uncharted in cryptographic hash constructions. Most existing lightweight hash functions depend on substitution permutation networks or a custom permutation structures. Furthermore, the conventional structure like the Merkle–Damgård is susceptible to length extension attack. This results to reduced security margin when the internal state size is limited. On the other hand, the sponge construction function introduces a flexible parameter that balances throughput and security to address these weaknesses. But, recent studies have shown that even sponge-based construction functions exhibit vulnerabilities when internal state sizes are insufficiently large [10]. Improving the internal state security of hash construction functions has turned out to be a significant research direction. One potential method is the wide pipe hashing construction function, which increases the internal state size of the compression function relative to the output digest size to strengthen resistance against collision attacks. Consequently, integrating wide pipe construction function with ARX – based encryption is largely unexplored for resource – constrained environment. Several important challenges remain unresolved despite significant progress made in lightweight cryptography. First, most lightweight cryptographic research focused on block ciphers, deserting the development of lightweight hash constructions. Second, existing lightweight hash functions often rely on specialized permutation structures, which increases implementation complexity. Third, Speck, which is an ARX-based lightweight block cipher, delivers an excellent performance for embedded systems. Finally, integrating the wide pipe construction function with Speck has not been extensively investigated. To mitigate these constraints, this paper introduces hSpeck-WidePipe, an innovative lightweight cryptographic hash function for resource-constrained applications. The combined ARX-based from the Speck lightweight algorithm and internal state security of WidePipe hashing construction provide efficiency and enhanced security. This will address the balanced trade-off between performance, deployment, and tamper resistance optimized for IoT and embedded systems.

Contributions

This study offers the following contributions:

- Design, development, and integration of ARX-based and WidePipe construction as a lightweight cryptographic hashing algorithm.
- Performance benchmarking and microarchitectural assessment using Linux perf profiling tool to evaluate efficiency, overhead, and hardware metrics.
- Exhaustive statistical analysis of randomness to assess diffusion, distribution, and pseudo random properties of the generated hash outputs using Dieharder battery test and NIST Statistical Test Suite.
- Suitability demonstration by highlighting the potential of hSpeck-WidePipe for IoT deployment and security applications on embedded systems.

II. LITERATURE SURVEY

The emergence of Internet of Things (IoT) has dramatically increased the number of interconnected devices. These often involve thousands to billions of distributed nodes collecting and transmitting sensitive data across potentially insecure transmission media. The security of these systems relies heavily on cryptographic mechanisms. However, many embedded devices overwhelm by conventional cryptographic protocols designed for high performance computing systems. These devices have become a high value targets for data manipulation, eavesdropping, and unauthorized access [11]. The adoption of conventional algorithms may lead to excessive latency, increased energy consumption, or even system failure because IoT nodes typically operate on limited energy sources [12]. IoT architectures introduce unique security requirements because of distributed and decentralized nature due to performance limitations. This increases vulnerability to attacks due to remote locations of the devices. Furthermore, blockchain technology relies heavily on cryptographic hash functions to ensure data integrity and immutability. Therefore, the effectiveness depends on hash algorithms that can operate efficiently while maintaining robust security properties [13]. These challenges necessitate a need to achieve an optimal balance between security robustness and computational efficiency through efficient cryptographic designs. A cryptographic hash function is the foundation of information systems security. It transforms an input message of arbitrary length into a fixed sized message digest. It plays a pivotal role in database indexing and data retrieval. The hash function establishes a framework for detecting unauthorized data modification and verifying the authenticity of transmitted information [14]. Therefore, cryptographic hash function must adhere to key principles to be considered secure. Various cryptographic hash function design have been developed specifying construction techniques to generate a message digest. The Merkle–Damgård construction processes the input message through an iterative compression function. This makes the construction collision resistant and secured compression functions [15]. However, it is vulnerable to multicollision and length extension attacks. HAIFA and sponge construction were introduced to address these vulnerabilities. The HAIFA improves Merkle–Damgård construction by adding a counter and optional salt to strengthen security against length extension attacks. The sponge construction processes input message through absorbing and squeezing phases and operates at a large internal state. Improving flexibility and strengthening cryptographic security. The design further improves through WidePipe construction by increasing the internal state relative to the output digest. This enhances resistance against collision attacks while surpassing conventional birthday bound limitations. Therefore, the adoption of WidePipe construction function becomes a compelling approach to bolstering security resilience while preserving computational efficiency. Lightweight cryptography presents resource efficient data protection to meet rising security needs for the Internet of Things and embedded system environments.

Applying a conventional security algorithm can lead to performance overhead. Lightweight algorithms are designed on basic mathematical operations, reduced key schedules, and compact memory requirements [2]. This allows the algorithm to be deployed in resource – constrained devices. In addition, lightweight cryptographic protocols are used in blockchain technologies and distributed sensor networks [16]. The ARX-based approach is one of the many approaches to lightweight cryptographic design. This has attracted widespread interest due to simplicity and efficiency relying on addition, rotation, and xor operations. The Speck cipher is a notable example designed specifically for software efficiency. Speck has undergone extensive scrutiny on embedded systems and Internet of Things [17]. Several hash functions offering different architectural designs and strategies to enhance performance and security trade-off have been introduced. These have been evaluated based on computational efficiency, cryptanalysis resistance, memory footprints, and scalability. These development emphasizes hash function landscape that focuses on balancing security robustness and performance efficiency. Performance optimization of hash functions continues to be a cornerstone in cryptographic research. Enlarging the internal state of the WidePipe construction enhances security without compromising performance. These optimization techniques underscore how innovative cryptography simultaneously boosts efficiency and security. Several issues remain open despite substantial progress in cryptographic hash function design. Finding a hash algorithm that is both secure and fast enough for limited hardware is still a major challenge. ARX – based Speck application within a hash construction framework is relatively underexplored. Likewise, WidePipe architecture shows resistance to collision attack. However, many lightweight designs underutilize this structural advantage. Moreover, resource – constrained devices continue to demand for more strong security assurance while limiting energy footprints and computational overhead. These drawbacks uncover an empirical research gap. The need for a hash function that combines ARX – based and the WidePipe construction function which could immensely enhance both performance and security. To address these challenges, hSpeck-WidePipe is introduced by combining Speck's ARX-based efficiency with the robust security properties of WidePipe architecture. Integrating this design allows the proposed method to deliver high performance and secure hashing within the resource-constrained environments.

III. MATERIALS AND METHODS

This section presents the tools and step-by-step approach used to design, develop, and evaluate the performance and security properties of the new hSpeck-WidePipe hashing algorithm. It presents the software, hardware, and datasets employed. Followed by the thorough discussion of the algorithm's design, implementation, and evaluation. Performance profiling was conducted on software, hardware, and statistical analyses of randomness. Time and space complexity was examined to identify the algorithm's behavior and computational efficiency.

Materials

Software

The new hashing function was developed in Python (PyCharm IDE, Windows 11 64-bit). The algorithm's randomness was validated using NIST STS SP 800-22 rev1a and Dieharder Battery Test suites. While the perf on Kali Linux was utilized to profile the performance.

Hardware

The following devices and specifications are used to evaluate the hardware performance of the new hSpeck-WidePipe hashing algorithm. These devices were selected to represent varying computational capabilities, memory capacities, and communication features. These enable comprehensive assessment of the algorithm's efficiency, execution time, and resource utilization in resource - constrained environments.

Arduino Uno

The Arduino Uno is powered by an 8-bit ATmega328P processor operating at 16 MHz with a 5V operating voltage. The board features 32 KB Flash memory and 2 KB of SRAM. Providing I/O connectivity for sensor interfacing and a USB-B interface for program loading.

Arduino Mega

The Arduino Mega 2560 for complex projects requiring extensive pins and robust memory. It supports 54 digital input/output pins, 16 analog inputs and operating at 5V. The memory resources includes 256 KB flash, 8 KB SRAM, and 4 KB EEPROM.

NodeMCU ESP 8266

The ESP8266 is a cost effective high performance Wi-Fi microcontroller designed for Internet of Things applications. It features a 32-bit processor running at 80 MHz expandable to 160 MHz. A 4 MB Flash memory combined with 160 KB RAM. Built-in 2.4 GHz Wi-Fi, which supports secure WPA/WPA2 protocols.

RC522

The MFRC522 is a 40x60mm RFID reader/writer module that operates at 13.56 MHz. It is ideal for low power IoT and smart device applications. It enables contactless communication with passive RFID tags within a short range of 0–60 mm.

Data

A total of 150 million hashed values were generated and tested for the statistical analysis of randomness, where 50 million hashed values from each variant were used as input.

Methods Design

This subsection presents the cryptographic design and operational flow of the new hSpeck-WidePipe hashing algorithm. It defines the fusion of kSpeck and WidePipe construction for an enhanced cryptographic model. The formal algorithmic description specifies the systematic procedures that establish a clear and reproducible foundation for both implementation and evaluation.

Cryptographic Model of the hSpeck-WidePipe Construction

The hSpeck-WidePipe hashing function integrates the Speck block cipher as permutation primitive within a WidePipe construction function. It provides computational efficiency without compromising security designed for resource – constrained devices.

Let:

E_K = kSpeck encryption function under key K
 P = plaintext block
 $C = E_K(P)$ ciphertext result
 S = random salt
 $\parallel$ = concatenation

The hash output M is constructed by prepending the salt to the ciphertext. This ensures domain separation and boost entropy as the internal state is larger than the final output following the WidePipe paradigm. This increases the security margin to limit generic attacks.

$$M = S \parallel C \quad (1)$$

Let $State = (s_0, s_1, \dots, s_{n-1})$ be the internal state of size n bits, partitioned into two logical components:

R = rate portion that absorbs input blocks
 C_p = capacity portion that provides security margin
 Thus,

$$State = R \parallel C_p \quad (2)$$

The message M is partitioned into blocks matching the rate width, with each block mixed into the state via XOR and subsequent kSpeck permutation:

$$State_i = \pi(State_{i-1} \oplus M_i) \quad (3)$$

After the final block is absorbed and padding is applied using a single-bit scheme. A final permutation π constructed from repeated applications of the kSpeck encryption function. This ensures full diffusion of the capacity portion into the rate portion. The final hash value H is obtained by truncating the rate component $H = Trunc_h(State)$ where h is the desired hash length. The design follows the transformation:

$$H = \text{kSpeckWidePipe}_K(S \parallel E_K(P)) \quad (4)$$

hSpeck-WidePipe Pseudocode

Algorithm 1: hSpeck-WidePipe Hash Construction

Input:

P : plaintext block
 K : secret key
 S : random salt

Output:

H : hash digest

```

1:  $C \leftarrow \text{kSpeck\_Encrypt}(K, P)$ 
2:  $M \leftarrow S \parallel C$ 
3: Initialize state  $State \leftarrow 0$ 
4: Split  $M$  into  $r$ -bit blocks  $M_1, M_2, \dots, M_t$ 
5: for  $i = 1$  to  $t$  do
6:    $State\_rate \leftarrow State\_rate \text{ XOR } M_i$ 
7:    $State \leftarrow \text{kSpeck\_Permutation}(State)$ 
8: end for
9: Apply padding: 0x80 followed by zeros
10:  $State \leftarrow \text{kSpeck\_Permutation}(State)$ 
11:  $H \leftarrow \text{Truncate}(State, h)$ 
12: return  $H$ 

```

Implementation

The new hSpeck-WidePipe hashing function was deployed and integrated on Arduino Uno, Mega, and ESP8266 platforms paired with RC522 RFID modules. The cryptographic model was translated into operational and resource-constrained prototype. The implementation was optimized to remain computationally feasible by adjusting the memory on Arduino Uno, state optimization on Arduino Mega, and wireless capabilities leveraging for ESP8266. The RC522 RFID module was used as the input component. The RC522 sent the UID or information to the microcontroller after a tag was detected. The received data was preprocessed, formatted into fixed blocks, padded, and sent to the hash function. The message digest which could be used for verification, checking, or storage is now the secure hashed.

Evaluation

This study assesses the performance, efficiency, and cryptographic robustness of the proposed hSpeck-WidePipe hashing construction. It integrates empirical and analytical approaches. This ensures that the algorithm satisfies functional requirements meeting established standards for security, efficiency, and resource-constrained environments implementation feasibility [18], [19].

Software Performance

The performance of hSpeck-WidePipe was analyzed using Kali Linux's perf utility. This focuses on retiring, bad speculations, frontend bound, backend bound, and instructions per cycle as evaluation metrics. Retiring refers to the final stage of the CPU pipeline where instructions are executed and completed. High retiring percentage indicates high efficiency. Bad speculations indicate that the CPU performed discarded work due to incorrect speculative execution. A low bad speculation percentage signifies a stable and efficient state. Frontend bound measures how frequently the CPU is starved for instructions. This is due to inability to decode instruction supply of the execution units. Backend bound indicates that CPU pipelines are stalled due to resource absence. A high backend bound signals that the processor has instructions ready to execute but is bottlenecked by memory latency. The instruction per cycle (IPC) measures the number of instructions finished by the CPU per clock cycle, as presented in (5); higher IPC means better performance and CPU utilization.

$$IPC = \frac{\text{Instructions Retired}}{\text{CPU Cycles}} \quad (5)$$

Hardware Performance

The hardware implementation of hSpeck-WidePipe was assessed based on execution time (ms), throughput (kB/s), and energy (μJ). The execution time measured in milliseconds (ms) is the time required to produce a hashed value. A lower value indicates a faster algorithm. The throughput as presented in (6) measures the data processing rate of the algorithm in kilobytes per second (kB/s). A higher throughput value indicates better performance.

$$\text{Throughput} = \frac{\text{Data Size}}{\text{Execution Time}} \quad (6)$$

The energy (μJ) indicates the battery power required by the hashing function to complete the task as expressed in (7). An efficient algorithm reflects low power consumption.

$$\text{Energy}(\mu\text{J}) = \text{Power (W)} \times \text{Execution Time (ms)} \times 10^6 \quad (7)$$

Statistical Analysis

Statistical analysis of randomness validates the cryptographic strength of the proposed construction. The Dieharder battery test and NIST Statistical Test Suite (STS) are standard test suites utilized in this paper. These evaluate the quality of the hash outputs in terms of uniform distribution, independence, and unpredictability. These are essential for resisting cryptographic attacks such as collision and preimage attacks. The evaluation begins with the generation of at least 1,000,000 bits binary sequences derived from the hSpeck-WidePipe hashing function exported in .bin.

Dieharder Battery Test

The DieHarder v3.31.0 evaluates randomness based on p-value distribution rather than providing a definitive status. A result between 0.025 and 0.975 is considered statistically pass. P-values converging at 0 or 1 are classified as fail, while on the periphery of these bounds are flagged as Weak.

NIST Statistical Test Analysis

Each test produces p-values, proportion of passing sequences, and uniformity of distribution. The test executed a full suite of 15 statistical test using the standard configuration parameters: $\alpha = 0.001$. The interpretation of results follows proportion test (8) and uniformity test. A p-value of p-values ≥ 0.001 indicates acceptable uniformity.

$$P_{\text{expected}} \pm 3 \sqrt{\frac{P(1-P)}{n}} \quad (8)$$

where:

$$P = 1 - \alpha$$

n = number of sequences

A strong compliance with NIST criteria confirm pass rates > 98.05%. All the p-values across tests and sequences are pooled to form a distribution by variant. A bloxplot was used to summarize the distribution of p-values for each variant.

Time and Space Complexity

The computational cost of the hSpeck-WidePipe was determined using time and space complexity. The Big O Notation, memory footprint, and efficiency trade-offs evaluates the hash functions asymptotic behavior. These provide comprehensive understanding on scalability and practicality deployment of hSpeck-WidePipe for resource-constrained environment.

Time Complexity

The time complexity analysis helps determine the predictability and scalability of hSpeck-WidePipe performance across output sizes. Each experiment was performed five times and the average execution time was computed. This is to minimize the effects of system noise and transient fluctuations. The $O(n)$ time complexity implies constant rate increase relative to the output size growth. The slope refers to the required time while the intercept is the fixed overhead. Linear regression (9) was used to model the relationship between output size and execution time. The regression model and goodness of fit metric assures the statistical and experimental validation.

$$T(n) = slope(n) + intercept \quad (9)$$

where:

$T(n)$ = execution time

n = output size

The observed execution time scales with the output size confirms the coefficient of determination R^2 indicating the goodness of fit. This guarantees deployment suitability in embedded systems of the new algorithm.

Space Complexity

The predictability and scalability on memory utilization of hSpeck-WidePipe was verified through empirical space complexity analysis. The stack allocation, heap usage, and any auxiliary buffers utilized during runtime concludes the total memory consumption. The average memory usage from the five runs was recorded to ensure consistency and minimize noise measurement. The relationship between output size and memory usage was analyzed through linear regression (10).

$$S(n) = slope(n) + intercept \quad (10)$$

where:

$S(n)$ = memory usage

n = output size

The growth pattern through the determination coefficient R^2 validates the relationship and suitability for resource-constrained environment deployment. By combining experimental measurement, regression modeling, and statistical validation merits space complexity.

IV. RESULTS AND DISCUSSION

Software Performance

The Top-Down Microarchitectural Analysis (TMA) method as presented in the figure below evaluates the software performance of the hSpeck-WidePipe 128, 192, and 256 variants.

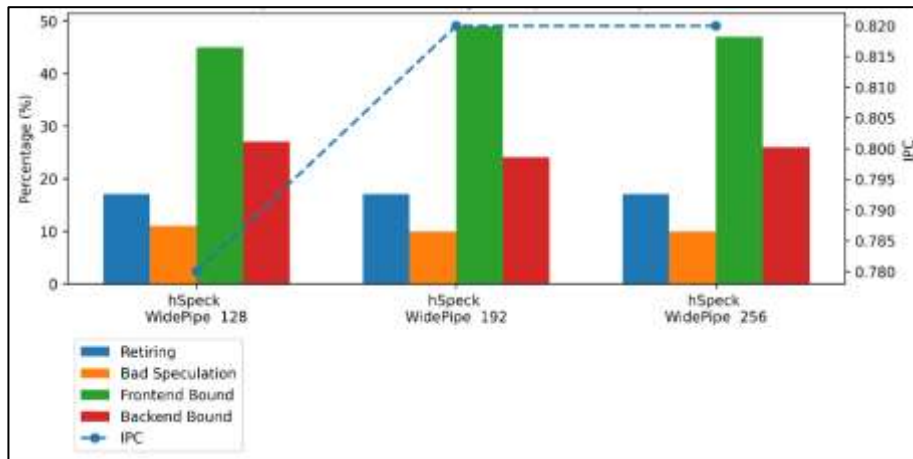


Figure 1: hSpeck-WidePipe Software Performance Comparative Analysis.
Source: Authors, (2026).

The microarchitectural distribution of hSpeck-WidePipe in Figure 1 reveals a relatively consistent execution profile across all three variants. The retiring component signifies uniform instruction at roughly 17% across all configurations regardless of parameter size. Bad speculation at $\approx 10\text{--}11\%$ is low due to negligible pipeline penalties from branch mispredictions or speculation failures. However, low speculative execution is advantageous as deterministic pathways prevent unexpected pipeline disruptions and ensure consistent execution time [20]. Frontend bound shows a dominant performance breakdown registering $\approx 45\%$, $\approx 50\%$, and $\approx 47\%$ in hSpeck WidePipe 128, 192, and 256, respectively. This indicate the processor delivers instructions to the execution engine rather than arithmetic or memory limitations The backend bound reflect moderate pressure on execution and memory contributing between $\approx 24\%$ and $\approx 27\%$. This do not impose excessive execution since backend bound stall is lower than frontend bound stall. Microarchitectural profiling shows that optimizing for frontend bottlenecks is crucial. This confirms performance is restricted by instruction delivery rather than memory latency or speculation failures [21]. The instructions per cycle (IPC) emphasizes the efficiency of the hashing function design. The hSpeck-WidePipe 128 variant achieves ≈ 0.78 , while hSpeck-WidePipe 192 and hSpeck-WidePipe 256 reach ≈ 0.82 . This suggest consistent improvement in sustained throughput for the larger variants. Using ARX – based operator runs efficiently on IoT processors without the need for complex control flow [8]. The hSpeck-WidePipe parameterization improves performance without altering pipeline stall distribution or introducing control flow complexity. This allows the processor to maintain a stable execution pattern and efficient instruction throughput. The design leverages the ARX approach to keep things simple.

Hardware Performance

This study considers execution time, throughput, memory usage, CPU cycles, and energy consumption as performances indicators. Cryptographic primitives must be tailored to strict energy, memory, and computational benchmarks to ensure viability on IoT devices [22], [23]. These are widely used metrics for assessing lightweight cryptographic algorithms in resource-constrained environment. Table 1 below presents the hardware performance of hSpeck-WidePipe variants evaluated on the Arduino Uno.

Table 1: Comparative hardware performance on Arduino Uno.

Algorithm	Execution Time (ms)			Throughput (kB/s)			Memory Usage (bits)			CPU Cycles (MHz)			Energy (μJ)		
	128	192	256	128	192	256	128	192	256	128	192	256	128	192	256
AESTiny-128	1.200	1.560	9.010	13.021	10.016	1.734	129	129	165	19200	24960	144160	420.00	546.00	3153.50
BLAKE2b	2.631	3.621	4.820	5.939	4.315	3.242	512	530	582	42096	57936	77120	920.85	1267.35	1687.00
BLAKE2s	1.631	3.024	3.670	9.580	5.167	4.257	268	268	302	26096	48384	58720	570.85	1058.40	1284.50
hSpeck-WidePipe 128	0.680	2.750	2.844	22.978	5.682	5.494	268	354	378	10880	44000	45504	238.00	962.50	995.40
hSpeck-WidePipe 192	0.760	2.580	2.920	20.559	6.056	5.351	367	373	401	12160	41280	46720	266.00	903.00	1022.00
hSpeck-WidePipe 256	0.844	3.980	3.008	18.513	3.926	5.194	524	534	553	13504	63680	48128	295.40	1393.00	1052.80
SHA-224	2.689	3.946	5.863	5.811	3.960	2.665	225	234	291	43024	63131	93808	941.15	1381.00	2052.05
SHA3-256	1.600	2.458	3.870	9.766	6.357	4.037	268	261	278	25600	39328	61920	560.00	860.30	1354.50
SHA3-512	2.810	3.560	5.970	5.560	4.389	2.617	396	521	591	44960	56960	95520	983.50	1246.00	2089.50
SHA-384	4.142	4.832	7.104	3.772	3.234	2.199	385	395	402	66272	77312	113664	1449.70	1691.20	2486.40
SHA-512	2.410	3.356	6.840	6.483	4.656	2.284	523	524	560	38560	53696	109440	843.50	1174.60	2394.00

Source: Authors, (2026).

The hSpeck-WidePipe variants offer substantial speed advantages over conventional hashing methods as shown in Table 1 above. hSpeck-WidePipe-128 requires only (0.680 ms) showing substantial performance gain over BLAKE2s (1.631 ms) and SHA3-256 (1.600 ms). The processing times for hSpeck-WidePipe-192 and -256 are (0.760 ms) and (0.844 ms), respectively. While SHA-224 (2.689 ms) and SHA-384 (4.142 ms) have higher computational overhead. These results highlights that the proposed algorithm achieves a 2–6 $\times$ speedup over conventional hashing functions on the Arduino Uno. The hSpeck-WidePipe-128 achieves a throughput of 22.978 kB/s. This is more than double the throughput of BLAKE2s (9.580 kB/s) and SHA3-256 (9.766 kB/s). The algorithm maintains better throughput ranging from 5.351 kB/s to 6.056 kB/s on 192 and 256 output sizes. This indicates a stable scalability across varying output lengths. The reduced execution latency directly contributes to improved throughput performance. These improvements is attributed to the ARX-based operations. Providing strong diffusion properties while maintaining efficient implementation [22]. The memory usage shows that hSpeck-WidePipe-128 requires 268 bits of memory comparable to BLAKE2s (268 bits) and SHA3-256 (268 bits). While significantly smaller than SHA3-512 (502.67 bits) and SHA-512 (523 bits). Efficient memory management is necessary in embedded systems to meet the simultaneous demands of networking, sensors, and application tasks [23]. The computational efficiency of hSpeck-WidePipe was attested by CPU cycle count. The hSpeck-WidePipe 128 requires only 10,880 CPU cycles. This is significantly lower than SHA-224 (43,024 cycles) and SHA-384 (66,272 cycles). Improved processing speed and reduced energy consumption directly minimizes instruction cycles. The new hashing function shows a substantial reduction in energy consumption over standard hash functions. This addresses the critical requirement of power efficiency in resource-constrained IoT devices. The hSpeck-WidePipe-128 operates at 238 μJ compared with BLAKE2s (570.85 μJ), SHA3-256 (560 μJ), and SHA-384 (1449.7 μJ). Even the hSpeck-WidePipe 192 and hSpeck-WidePipe 256, a higher security configurations achieved fairly low energy consumption of 266 μJ and 295 μJ respectively. The proposed hSpeck-WidePipe family exhibits a competitive balanced profile on the Arduino Mega implementation across latency, throughput, memory footprint, and energy consumption as shown in Table 2 below.

Table 2: Comparative hardware performance on Arduino Mega.

Algorithm	Execution Time (ms)			Throughput (kB/s)			Memory Usage (bits)			CPU Cycles (MHz)			Energy (μ J)		
	128	192	256	128	192	256	128	192	256	128	192	256	128	192	256
AESTiny-128	1.245	2.658	4.502	12.550	5.878	3.471	131	134	144	19920	42528	72032	435.75	930.30	1575.70
BLAKE2b	2.415	5.456	3.324	6.470	2.864	4.701	521	526	546	38640	87296	53184	845.25	1909.60	1163.40
BLAKE2s	1.635	3.654	1.634	9.557	4.276	9.562	265	298	301	26160	58464	26144	572.25	1278.90	571.90
hSpeck-WidePipe 128	1.654	4.130	3.650	9.447	3.783	4.281	265	365	375	26464	66080	58400	578.90	1445.50	1277.50
hSpeck-WidePipe 192	1.785	3.650	4.032	8.754	4.281	3.875	391	425	436	28560	58400	64512	624.75	1277.50	1411.20
hSpeck-WidePipe 256	2.354	4.560	4.568	6.638	3.427	3.421	516	564	571	37664	72960	73088	823.90	1596.00	1598.80
SHA-224	2.412	3.256	1.687	6.478	4.799	9.262	232	248	254	38592	52096	26992	844.20	1139.60	590.45
SHA3-256	1.620	3.623	2.687	9.645	4.313	5.815	264	265	268	25920	57968	42992	567.00	1268.05	940.45
SHA3-512	2.865	4.963	2.402	5.454	3.148	6.505	514	525	526	45840	79408	38432	1002.75	1737.05	840.70
SHA-384	4.012	4.102	2.326	3.895	3.809	6.718	385	395	411	64192	65632	37216	1404.20	1435.70	814.10
SHA-512	2.474	4.897	3.564	6.316	3.191	4.384	516	532	541	39584	78352	57024	865.90	1713.95	1247.40

Source: Authors, (2026).

The hSpeck-WidePipe 128 ends in 1.654 ms, 9.447 kB/s throughput, 265 bits memory, 26,464 cycles, and 578.90 μ J energy. Its performance is comparable to standard embedded baselines SHA-256 (1.620 ms; 9.645 kB/s; 264 bits; 25,920 cycles; 567.00 μ J) and BLAKE2s (1.635 ms; 9.557 kB/s; 265 bits; 26,160 cycles; 572.25 μ J). While surpassing heavier choices SHA-384 (4.012 ms; 3.895 kB/s; 385 bits; 64,192 cycles; 1,404.20 μ J) and SHA3-512 (2.865 ms; 5.454 kB/s; 514 B; 45,840 cycles; 1,002.75 μ J) on energy and cycles. The hSpeck-WidePipe 128 effectively delivers near SHA-256 throughput and energy. The hSpeck-WidePipe 192 records 3.650 ms, 4.281 kB/s, 425 bits memory, and 1,277.50 μ J. This remains competitive in energy to SHA-256 (1,268.05 μ J) and BLAKE2s (1,278.90 μ J). Staying far below SHA-384 (1,435.70 μ J) and SHA-512 (1,713.95 μ J). The hSpeck-WidePipe 128 at 256 bits tallied 3.650 ms with 4.281 kB/s at 1,277.50 μ J. This indicates that the design remains operationally efficient [24]. Using lightweight block ciphers in hash functions reduces energy and computational costs in constrained environments while keeping data secure [8]. In this context, hSpeck-WidePipe is promising not because it always tops a single metric. But because it stays close to SHA-256 and BLAKE2s while offering configurable WidePipe instantiations. The comparative hardware performance of the hSpeck-WidePipe hash variants on NodeMCU ESP8266 microcontroller operating at 80 MHz as depicted in Figure 2 below against widely used cryptographic algorithms.

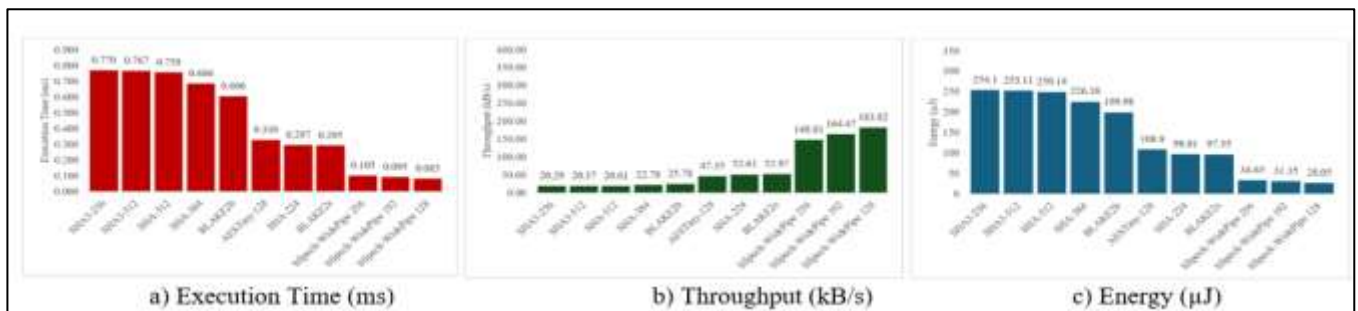


Figure 2: Comparative hardware performance on NodeMCU ESP 8266 where a) Execution Time, b) Throughput, and c) Energy Frequency = 80 MHz.

Source: Authors, (2026).

The hSpeck-WidePipe family shows a significant performance advantage over conventional hash algorithms. The -256, -192, and -128 variants in Figure 2 (a) achieving execution times of 0.105 ms, 0.095 ms, and 0.085 ms, respectively. While SHA3-256 (0.770 ms), SHA3-512 (0.767 ms), SHA-512 (0.758 ms), and SHA3-384 (0.686 ms) demonstrate higher overhead reflecting sponge based construction complexity. BLAKE2b offers minor performance gains (0.606 ms), AES(Tiny)-128 (0.330 ms), SHA-224 (0.297 ms), and BLAKE2s (0.295 ms). All remain quite slower than the proposed construction approach. These results represent a roughly 7–9 $\times$ efficiency over SHA3-256. Lightweight cryptographic are extensively implemented in IoT security research to minimize operational overhead and instruction complexity while maintaining robust security [25]. The execution time improvements directly enhance throughput as presented in Figure 2 (b). This addresses the low performance of conventional cryptographic hash functions. SHA3-256, SHA3-512, and SHA-512 deliver throughputs between 20–23 kB/s. BLAKE2b performs better at 25.78 kB/s. The lightweight variants SHA-224 and BLAKE2s achieved a faster throughput of 52 kB/s offering superior suitability for constrained environments. Furthermore, the proposed hSpeck-WidePipe variants tallied throughputs of 256-(148.81 kB/s), 192-(164.47 kB/s), and 128-(183.82 kB/s) achieving superior performance. Alleviating computational bottleneck and improving system responsiveness on resource-constrained devices can be achieved through efficient hashing [18]. The energy consumption results in Figure 2 (c) demonstrate the superior efficiency of hSpeck-WidePipe which requires significantly less power when compared with conventional hash functions. BLAKE2b (199.98 μ J) is more energy efficient compared to the 250 μ J consumed by SHA3-256, SHA3-512, and SHA-512. While the lightweight hashing functions AES(Tiny)-128, SHA-224, and BLAKE2s reduce the energy cost to 97–109 μ J. Contrary to hSpeck-WidePipe variants which require only 34.65 μ J, 31.35 μ J, and 28.05 μ J, 256-, 192-, and 128-respectively.

The proposed algorithm achieves a 6 to 9× reduction in energy consumption compared to conventional hash functions by lowering computational complexity and reducing required instruction cycles. Extending the operational lifespan of battery powered or energy harvesting IoT systems is vital to utilize energy efficient cryptographic primitives [25].

Statistical Analyses of Randomness

The statistical robustness of the proposed hSpeck-WidePipe variants were evaluated using the Dieharder battery test suite and NIST STS SP 800-22. These are well established framework for assessing the quality of pseudo-random number generators through multiple stringent statistical tests.

Dieharder Battery Test

The visualizations in Figure 3 – Figure 5 illustrates the comprehensive p-value randomness distributions for the Dieharder battery tests.

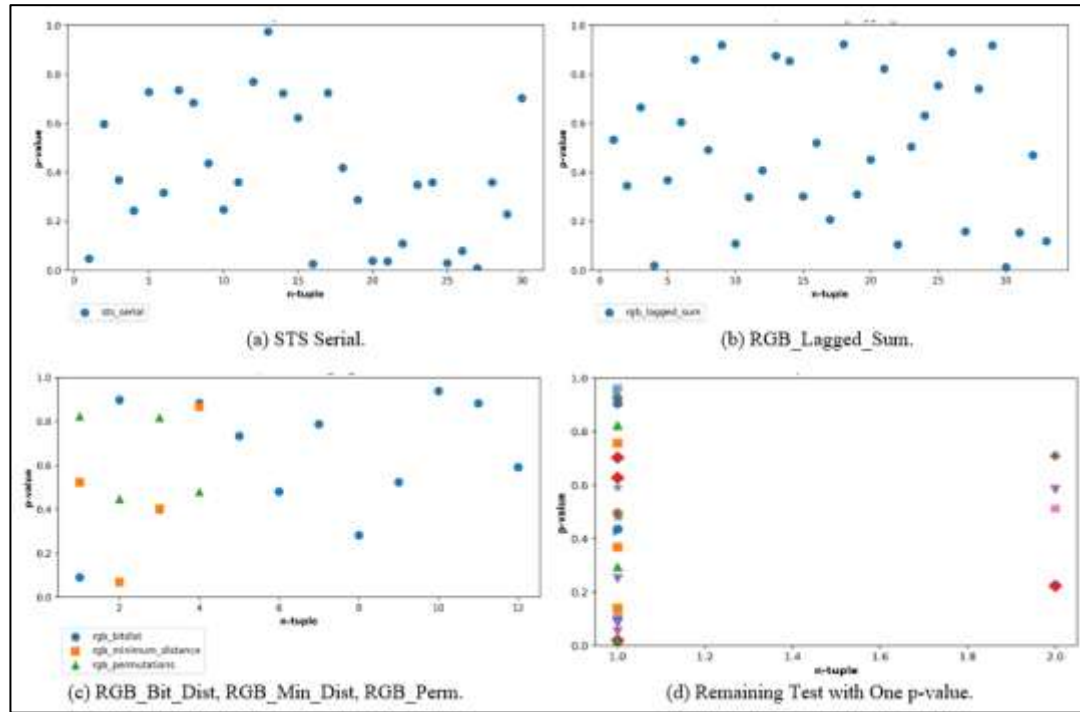


Figure 3: Dieharder battery test result of hSpeck-WidePipe 128 variant.

Source: Authors, (2026).

Several n-tuples occupy the lower tail as observed in Figure 3 (a) STS Serial. This does not automatically invalidate the results in a multi test scenario rather suggests potential persistent serial dependencies. However, this requires reevaluation to avoid over interpreting [26], [27]. The RGB_lagged_sum in Figure 3 (b) series lacks a monotone degradation pattern. This is interpreted as no persistent linear correlation signature suggesting that any dependencies are not tied to increasing tuple sizes. Moreover, additional independent streams should be conducted to confirm stability [28]. A relatively stable rgb_bitdist, but a single outlier hits rgb_minimum_distance in Figure 3 (c). This suggests that the test is sensitive to short range patterns rather than overall balance. Figure 3 (d) shows few low-tail exceptions and most p-values cluster are in the range of 0.3–0.9 and inside the non-rejection region. This indicates that hSpeck-WidePipe 128 shows no detectable bias. The majority of tests in hSpeck-WidePipe 128 exhibited moderate behavior reassures for IoT security frameworks. Poor randomness can severely compromise security protocols [29]. However, outliers in serial tests require stricter validation expected for cryptographic standards [30].

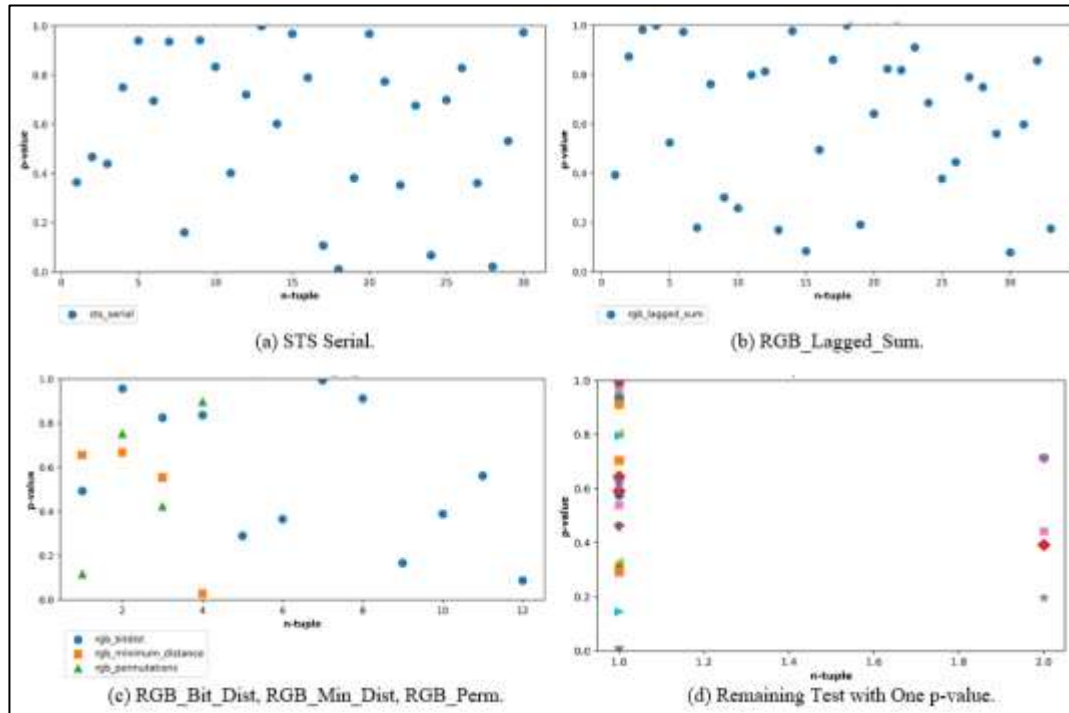


Figure 4: Dieharder battery test result of hSpeck-WidePipe 192 variant.

Source: Authors, (2026).

The STS Serial in Figure 4(a) shows that data is mostly random with only few specific n-tuples at low points. The RGB_Lagged_Sum test in Figure 4(b) shows middle range p-values with few very low p-values. This is typical when setting up lag-based tests. Nevertheless, the focus should be on whether low p-values are robust across different seeds and runs and not just one off extreme result. The RGB_BitDist points in Figure 4(c) remain generally considerable across n-tuples. This suggests secure behavior across overlapping tuple. A p-value of ≈ 0.03 on RGB_Minimum_Distance trial is hinting at a narrow sensitivity. However, the potential deviation appears to be localized effect rather than a broad failure since the remaining trials performed well. The remaining test with one-p-value summary in Figure 4 (d) indicates that most p-values fall within the acceptance region. Overall, the kSpeck-WidePipe-192 demonstrates strong statistical behavior under Dieharder battery test. However, using a consistent replication across different datasets is recommend to avoid misleading anomalies caused by test dependencies [31]. The p-values as presented in panels (a)–(d) of Figure 6 below across the Dieharder battery for hSpeck-WidePipe 256.

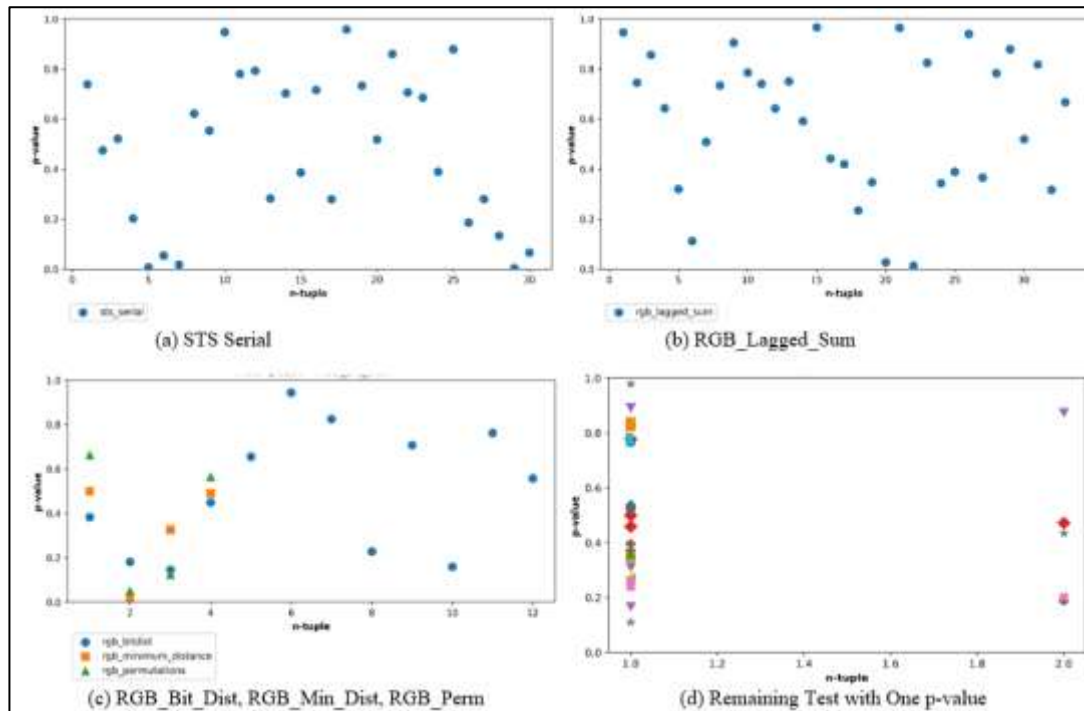


Figure 5: Dieharder battery test result of hSpeck-WidePipe 256 variant.

Source: Authors, (2026).

The STS Serial in Figure 5(a) shows a wide spread of few low tail points but many mid to high p-values. This means that STS serial pattern statistics are not violated when taken together. Which is important because serial type tests are sensitive to short range dependence. A largely upper midrange p-values is shown in the RGB_Lagged_Sum in Figure 5(b) with one or two near threshold points occurring at particular lag indices. Isolated low results may be incidental false positive errors rather than systemic issues [32]. It should not be interpreted as definitive failure without consistent replication rather requires further validation. The p-values in Figure 5(c) RGB_Bit_Distribution, RGB_Minimum_Distance, and RGB_Permutations are mostly above rejection thresholds with only few points near the lower tail. The consistency of low tail patterns indicates that kSpeck-WidePipe 256 maintains a statistically uniform bit distribution. These results align with recent empirical random number generator (RNG) validation study using Dieharder [33]. Finally, most test outputs in Figure 5(d) Remaining Test with One p-value concentrate in roughly midrange p-values. Small number of higher p-values appear occasionally. Occasional glitches during parallel testing are expected and is not a signs of a non randomness. Instead understand the standard practice of evaluating randomness using a full battery of tests rather than relying on single result [34]. As a result, Dieharder tests suggest hSpeck-WidePipe variants produce outputs that behave randomly as per standard [35].

NIST Statistical Test Analysis

The statistical randomness properties of hSpeck-WidePipe family were also evaluated using the NIST Statistical Test Suite as depicted in Figure 6 below through a boxplot distributions of p-values.

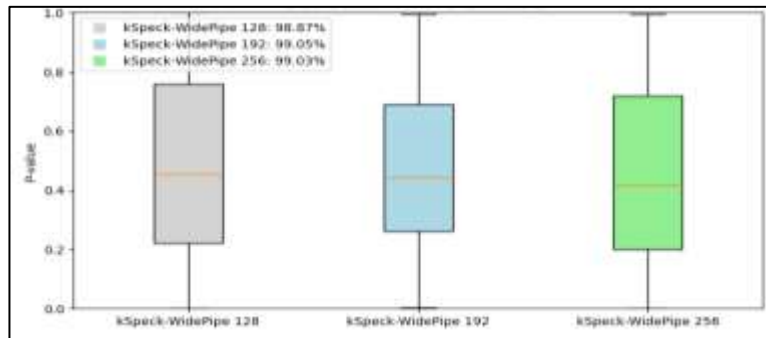


Figure 6: NIST randomness test result of hSpeck-WidePipe variants.
Source: Authors, (2026).

The hSpeck-WidePipe variants show a broad distribution across the interval centering around midrange values. This confirms that the generated sequences do not display significant structural bias. The wide spread in the data shows it behaves like true random samples. This confirms that the p-values generated are truly random sequences [36]. With pass rates exceeding 98.8% across all three versions suggests that hSpeck-WidePipe effectively satisfies the statistical requirements of the STS suite. High quality random number generators generally achieve pass proportions near 1.0 across the NIST SP 800-22 test suite including frequency and linear complexity while remaining within the prescribed confidence intervals [37]. With this, the hSpeck-WidePipe construction is stable achieving high pass rates across all settings. Increasing the state size from 128 to 256 bits keeps the randomness strong, as shown by consistent results across all tests [38].

Times and Space Complexity Analyses

Time Complexity

The empirical time complexity of hSpeck-WidePipe was examined on execution time scales with increasing output size as presented in Figure 8 below.

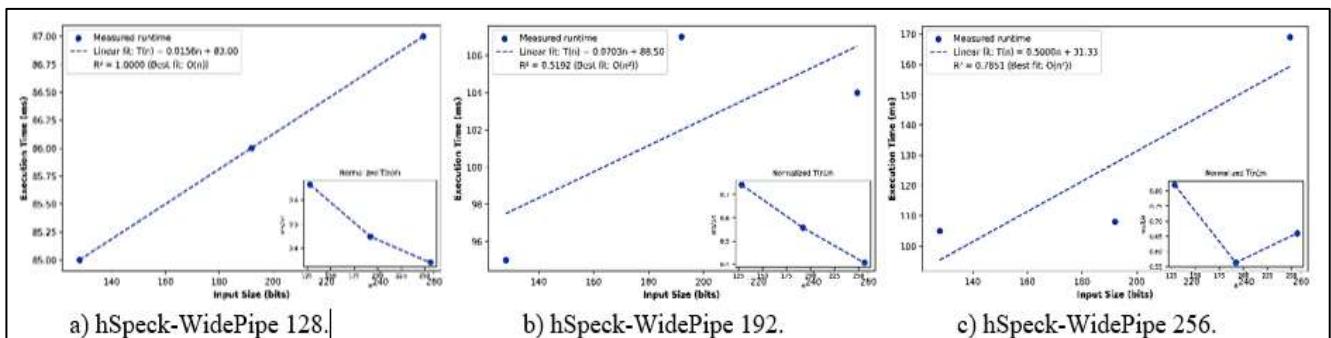


Figure 7: Empirical Time and Normalized Complexity of hSpeck-WidePipe variants.
Source: Authors, (2026).

The measured execution time in Figure 7 (a) increases from ≈ 85 ms to ≈ 87 ms as the output size grows. Linear regression $T(n)=0.0156n+83.00$ yields a perfect fit $R^2=1.0000$ confirming linear scaling with output length. The ≈ 83 ms intercept pointing to initialization overhead while ≈ 0.0156 slope demonstrates minimal computational cost. This is supported by decreasing normalized $T(n)/n$ metric showing that increased output processing reduces the cost per bit. For hSpeck-WidePipe 192 in Figure 7 (b), inconsistent timing data leads to a weak model fit $R^2=0.5192$ for the equation $T(n)=0.0703n+88.50$.

The weak correlation indicates that external factors largely shape the measurements rather than the main trend. Nevertheless, larger outputs improve the efficiency of lightweight algorithms by reducing the impact of initial setup costs on total execution time [39]. hSpeck-WidePipe 256 grows the fastest. Its regression model $T(n)=0.5000n+31.33$, $R^2=0.7851$ has a steeper slope than 128- and 192-bits as shown in Figure 7 (c). The empirical results still indicate that the runtime component scales with output length. The lower intercept ~ 31 ms compared with hSpeck-WidePipe 192- suggests that the fixed initialization overhead measured may differ due to implementation or measurement conditions. The cost per byte drops initially and then flattens out as input size grows. This shows that the algorithm handles larger data more efficiently. This behavior matches the expected performance of wide-pipe hashing structures [40].

Space Complexity

The memory usage of the hSpeck-WidePipe models based on output sizes is presented Figure 8 below.

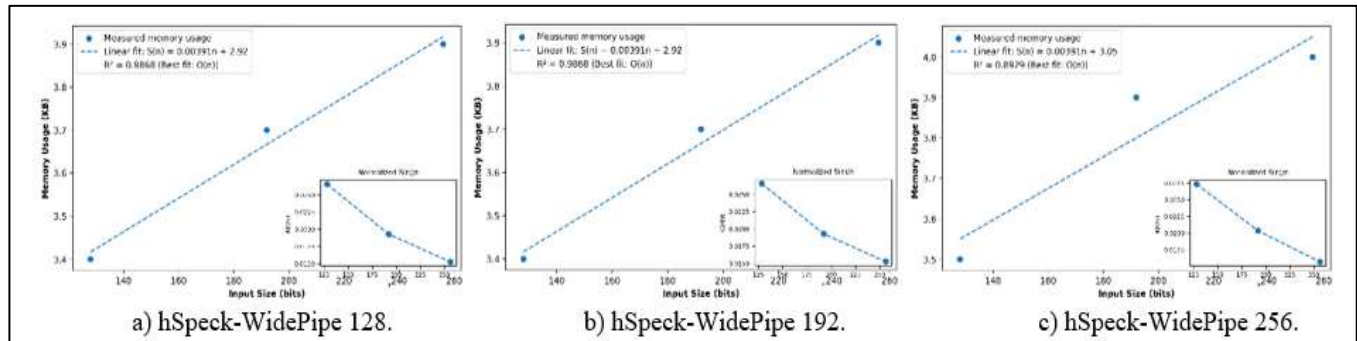


Figure 8: Empirical Space and Normalized Complexity of hSpeck-WidePipe variants.

Source: Authors, (2026).

The hSpeck-WidePipe 128 memory utilization scales from 3.4 KB to 3.9 KB as output size increases. The memory use increases at a very consistent and predictable rate as output grows based on the regression model $S(n) = 0.00391n + 2.92$ with a strong $R^2 = 0.9868$. The intercept = 2.92 KB refers to the memory used when the input size is zero. The 192 bit version behaves similarly to the 128 bit $S(n)=0.00391n + 2.92$ and $R^2=0.9868$. This is a clear indication of the hSpeck-WidePipe architecture memory usage and not the key size. Because the slope and intercept are the same. This predictable scaling where memory and processing power are tight is great for low power technologies like IoT devices [13]. The memory usage grows almost linearly for the 256-bit configuration, similar to other configurations (slope = 0.0039). The 3.05 KB intercept indicates a slightly higher baseline memory cost. The lower accuracy is probably due to inconsistent testing and not limited sample size. Rather than an increased complexity of the security protocol. Basically, the system may be too sensitive to environment changes and how it behaves at that moment [41]. Nevertheless, the normalized $S(n)/n$ values continue to decrease as the output size increases.

V. CONCLUSION

This study introduced a new lightweight cryptographic hash construction function that integrates the Speck block cipher and the WidePipe architecture. The design ensures a balance of low overhead and high security tailored for resource-constrained environments. The result demonstrated that combining Speck with widepipe structure increases the internal state size and improves security resilience. This solution reduced memory footprint and computational overhead making it ideal for resource-constrained devices. The modified hashing function demonstrated strong pseudo random behavior by passing both Dieharder and NIST SP 800-22 statistical analysis of randomness. The hSpeck-WidePipe achieved competitive performance throughput. Also, delivering a near linear processing time as the output data size increases. The optimized memory utilization guarantees superb space efficiency. The new algorithm assures suitability for real time applications for battery operated devices through minimal latency. This solution provides secure data integrity for environments with limited energy and computational power.

VI. LIMITATIONS AND FUTURE WORKS

This study relies on empirical security analysis rather than extensive formal cryptanalysis. The implementation of the new hashing function focuses only on specific hardware platforms. Future research will focus on performance acceleration testing via parallel hardware FPGA and ASIC integrated into IoT security architectures. Strengthening of security through cryptanalysis. Conduct benchmarking against NIST standard hash functions. Finally, investigate post quantum resilience.

VII. AUTHOR'S CONTRIBUTION

Conceptualization: Roman Alex F. Lustro and Daniel C. Neri.

Methodology: Roman Alex F. Lustro and Daniel C. Neri.

Investigation: Roman Alex F. Lustro and Daniel C. Neri.

Discussion of results: Roman Alex F. Lustro and Daniel C. Neri.

Writing – Original Draft: Roman Alex F. Lustro and Daniel C. Neri.

Writing – Review and Editing: Roman Alex F. Lustro and Daniel C. Neri.

Resources: Roman Alex F. Lustro and Daniel C. Neri.

Supervision: Roman Alex F. Lustro and Daniel C. Neri.

Approval of the final text: Roman Alex F. Lustro and Daniel C. Neri.

VIII. REFERENCES

- [1] M. Khan, D. Johansen, and H. Dagenborg, "A Comparative Analysis of Lightweight Hash Functions Using AVR ATXMEGA128 and ChipWhisperer," Aug. 2025, [Online]. Available: <http://arxiv.org/abs/2508.07840>
- [2] R. Iqbal, N. M. Ansari, M. ur R. Awan, M. Ismail, and H. Gul, "Design and Evaluation of Lightweight Cryptographic Algorithms for Internet of Things (IoT) Devices: Achieving Optimal Trade-Offs Between Security, Computational Speed, and Energy Efficiency in Resource-Constrained Environments," *THE PROGRESS: A Journal of Multidisciplinary Studies*, vol. 6, no. 1, pp. 85–99, Mar. 2025, doi: 10.71016/tp/smfybz24.
- [3] N. Sharma, H. Parveen Sultana, R. Singh, and S. Patil, "Secure Hash Authentication in IoT based Applications," in *Procedia Computer Science*, Elsevier B.V., 2019, pp. 328–335. doi: 10.1016/j.procs.2020.01.042.
- [4] N. F. Mufidah and H. H. Nuha, "Performance and Security Analysis of Lightweight Hash Functions in IoT," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 9, no. 3, pp. 264–270, Dec. 2024, doi: 10.30591/jpit.v9i3.7633.
- [5] M. Shashwat and S. B. Tech, "A Comprehensive Study of Cryptographic Hash Functions," 2024.
- [6] K. Sakan, K. Algazy, N. Kapalova, and A. Varennikov, "Lightweight Hash Function Design for the Internet of Things: Structure and SAT-Based Cryptanalysis," *Algorithms*, vol. 18, no. 9, Sep. 2025, doi: 10.3390/a18090550.
- [7] J. Kaur, A. C. Canto, M. M. Kermani, and R. Azarderakhsh, "A Survey on the Implementations, Attacks, and Countermeasures of the NIST Lightweight Cryptography Standard: ASCON," *ACM Comput. Surv.*, vol. 58, no. 1, Aug. 2025, doi: 10.1145/3744640.
- [8] A. Sevin, Ü. Çavuşoğlu, and Ç. Ş. Glu, "Design and Performance Analysis of a SPECK-Based Lightweight Hash Function," *Electronics (Basel)*, vol. 13, no. 23, pp. 1–22, 2024, doi: <https://doi.org/10.3390/electronics13234767>.
- [9] Z. Feng, Y. Luo, C. Wang, Q. Yang, Z. Liu, and L. Song, "Improved Differential Cryptanalysis on SPECK Using Plaintext Structures."
- [10] C. Freitag, A. Ghoshal, and I. Komargodski, "Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions," *Journal of Cryptology*, vol. 39, no. 1, Jan. 2026, doi: 10.1007/s00145-025-09558-1.
- [11] Z. A. Haider, A. Zeb, A. K. M. M. Islam, T. Rahman, A. Arishi, and I. Ullah, "Enhancing IoT security with resource-efficient cryptography: A comprehensive review of lightweight and hybrid algorithms," Feb. 01, 2026, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2025.100861.
- [12] M. A. Caraveo-Cacep, R. Vázquez-Medina, and A. Hernández Zavala, "A review on security implementations in soft-processors for IoT applications," *Comput. Secur.*, vol. 139, Apr. 2024, doi: 10.1016/j.cose.2023.103677.
- [13] S. Dhar, A. Khare, A. D. Dwivedi, and R. Singh, "Securing IoT devices: A novel approach using blockchain and quantum cryptography," *Internet of Things (Netherlands)*, vol. 25, Apr. 2024, doi: 10.1016/j.iot.2023.101019.
- [14] O. Kuznetsov, O. Peliukh, N. Poluyanenko, S. Bohucharskyi, and I. Kolovanova, "Comparative Analysis of Cryptographic Hash Functions in Blockchain Systems," Oct. 2023.
- [15] J. B. Damgård, "A Design Principle for Hash Functions," 1990.
- [16] A. Alfrhan, T. Moulahi, and A. Alabdulatif, "Comparative study on hash functions for lightweight blockchain in Internet of Things (IoT)," *Blockchain: Research and Applications*, vol. 2, no. 4, Dec. 2021, doi: 10.1016/j.bcr.2021.100036.
- [17] M. El Hanine, A. El-Yahyaoui, and R. Es-Sadaoui, "Design and assessment of lightweight cryptographic algorithms on ESP32 and STM32 for IoD security," *Egyptian Informatics Journal*, vol. 32, Dec. 2025, doi: 10.1016/j.eij.2025.100818.
- [18] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in IoT networks: A survey," *Future Generation Computer Systems*, vol. 129, pp. 77–89, 2022, doi: <https://doi.org/10.1016/j.future.2021.11.011>.
- [19] A. Rukhin, J. Soto, and J. Nechvatal, "Nistspecialpublication800-22R1a.Pdf," *Nist Special Publication*, vol. 22, no. April, pp. 1-1-G-1, 2010, [Online]. Available: <http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf>
- [20] X. Hu *et al.*, "Towards On-the-Fly Code Performance Profiling," *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 1, pp. 1–25, Jan. 2026, doi: 10.1145/3725212.
- [21] J. Wang, Z. Wang, W. (L.), and Y. Hu, "Optimizing the micro-architectural performance of the current and emerging edge infrastructure," *IEEE Transactions on Cloud Computing*, vol. 12, no. 1, pp. 40–52, 2024, doi: <https://doi.org/10.1109/TCC.2023.3333813>.
- [22] D. J. Bernstein *et al.*, "Gimli: a cross-platform permutation," 2017. [Online]. Available: <https://www.ams.org/profession/leaders/>
- [23] E. GAABOURI Ismail, S. Mohamed, and B. Mostafa, "A Survey on Lightweight Cryptography Approach For IoT Devices Security," 2022.
- [24] Y. B. Kim, T. Y. Youn, and S. C. Seo, "Chaining optimization methodology: A new sha-3 implementation on low-end microcontrollers," *Sustainability (Switzerland)*, vol. 13, no. 8, Apr. 2021, doi: 10.3390/su13084324.
- [25] F. M. Agugliaro *et al.*, "Academic Editors: Esther Salmerón-Manzano and A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers," 2024, doi: 10.3390/technologies.
- [26] K. Bhattacharjee and S. Das, "A search for good pseudo-random number generators: Survey and empirical studies," *Comput. Sci. Rev.*, vol. 45, no. 10047, 2022, doi: <https://doi.org/10.1016/j.cosrev.2022.100471>.
- [27] H. Chai, Q. Pan, and J. Wu, "A survey of random number generator: Approaches, tests, novel applications in block-chain and AI driven industrial networks," *Security and Safety*, vol. 4, p. 2025015, 2025, doi: 10.1051/sands/2025015.

- [28] L. Crocetti, P. Nannipieri, S. Di Matteo, L. Fanucci, and S. Saponara, "Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators," *Electronics (Switzerland)*, vol. 12, no. 3, 2023, doi: 10.3390/electronics12030723.
- [29] J. P. Hughes and W. Diffie, "Challenges Random Number Generators and in the Real World," 2022.
- [30] A. Kamadi and Z. Abbas, "Implementation of TRNG with SHA-3 for hardware security," *Microelectronics J.*, vol. 123, p. 105410, 2022, doi: <https://doi.org/10.1016/j.mejo.2022.105410>.
- [31] E. A. Luengo, B. A. Olivares, L. J. G. Villalba, and J. Hernandez-Castro, "Further analysis of the statistical independence of the NIST SP 800-22 randomness tests," *Appl. Math. Comput.*, vol. 459, Dec. 2023, doi: 10.1016/j.amc.2023.128222.
- [32] E. A. Luengo and L. J. G. Villalba, "Recommendations on statistical randomness test batteries for cryptographic purposes," *ACM Comput. Surv.*, vol. 54, no. 4, May 2022, doi: 10.1145/3447773.
- [33] A. Al-Daraiseh, Y. Sanjalawe, S. Al-E'mari, S. Fraihat, M. Bany Taha, and M. Al-Muhammed, "Cryptographic Grade Chaotic Random Number Generator Based on Tent-Map," *Journal of Sensor and Actuator Networks*, vol. 12, no. 5, Oct. 2023, doi: 10.3390/jsan12050073.
- [34] B. Ryabko, "Building Test Batteries Based on Analyzing Random Number Generator Tests within the Framework of Algorithmic Information Theory," *Entropy*, vol. 26, no. 6, Jun. 2024, doi: 10.3390/e26060513.
- [35] H. G. Ayoub, Z. A. Abdulrazzaq, A. F. Fathil, S. A. Hasso, and A. T. Suhail, "Unveiling robust security: Chaotic maps for frequency hopping implementation in FPGA," *Ain Shams Engineering Journal*, vol. 15, no. 11, Nov. 2024, doi: 10.1016/j.asej.2024.103016.
- [36] G. Liu, P. Mu, K. Wang, G. Guo, X. Liu, and P. He, "Random Numbers Generated Based on Dual-Channel Chaotic Light," *Electronics (Switzerland)*, vol. 13, no. 9, May 2024, doi: 10.3390/electronics13091603.
- [37] C. Foreman, R. Yeung, and F. J. Curchod, "Statistical Testing of Random Number Generators and Their Improvement Using Randomness Extraction," *Entropy*, vol. 26, no. 12, Dec. 2024, doi: 10.3390/e26121053.
- [38] V. Piscopo, A. Dolmeta, M. Mirigaldi, M. Martina, and G. Masera, "A High-Entropy True Random Number Generator with Keccak Conditioning for FPGA," *Sensors*, vol. 25, no. 6, Mar. 2025, doi: 10.3390/s25061678.
- [39] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., Jul. 2015. doi: 10.1145/2744769.2747946.
- [40] S. Lucks, "A Failure-Friendly Design Principle for Hash Functions," in *Advances in Cryptology - ASIACRYPT 2005*, Springer, 2005. doi: https://doi.org/10.1007/11593447_26.
- [41] A. Magyari and Y. Chen, "Securing the Internet of Things with Ascon-Sign," *Internet of Things (The Netherlands)*, vol. 28, Dec. 2024, doi: 10.1016/j.iot.2024.101394.