Jurnal Informatika: Jurnal pengembangan IT
Vol. 9, No. 3, 2024
ISSN : 2477-5126, DOI:10.30591/jpit.v9i3.7633

Performance and Security Analysis of Lightweight Hash Functions in IoT

Nada Fajri Mufidah, Hilal H. Nuha
School of Computing, Telkom University, Bandung, 40257, Indonesia

Info Artikel

Riwayat Artikel:

Received 2024-09-10
Revised 2024-10-14
Accepted 2024-10-17

Corresponding Author:

Hilal H. Nuha

Email:hilal@socj.telkomuniversity.ac.id

Abstract – *The rapid proliferation of Internet of Things (IoT) devices across various sectors, including healthcare, automotive, smart homes, and agriculture, has created a need for robust security measures that do not compromise the limited resources of these devices. This study analyses the performance and security of several lightweight cryptographic hash functions, specifically SHAKE128, BLAKE2s, SHA-256, SHA3-256, SipHash and xxHash, within the context of the Internet of Things (IoT). A series of experiments conducted on the Arduino Uno platform allows for an evaluation of these functions in terms of throughput, memory usage, and avalanche effect. The findings indicate that while SHAKE128 and SHAKE256 demonstrate superior throughput, they require greater memory, particularly with larger input sizes. BLAKE2s exhibits a robust equilibrium between throughput, memory efficiency, and consistent avalanche effects, rendering it a dependable option for 256-bit outputs. Conversely, xxHash and SipHash provide high throughput and minimal memory usage, yet exhibit reduced avalanche effects. The findings of this research provide critical insights for developers and researchers on the selection of appropriate cryptographic solutions, which must be tailored to the constraints and security requirements of IoT devices.*

Keywords: *hash function, IoT security, lightweight cryptography, memory efficiency, performance analysis*

Abstrak – *Perkembangan perangkat Internet of Things (IoT) yang cepat di berbagai sektor, termasuk kesehatan, otomotif, rumah pintar, dan pertanian, telah menciptakan kebutuhan akan langkah-langkah keamanan yang kuat yang tidak mengorbankan sumber daya yang terbatas pada perangkat-perangkat ini. Penelitian ini menganalisis kinerja dan keamanan beberapa fungsi hash kriptografi ringan, khususnya SHAKE128, BLAKE2s, SHA-256, SHA3-256, SipHash, dan xxHash, dalam konteks Internet of Things (IoT). Serangkaian percobaan yang dilakukan pada platform Arduino Uno memungkinkan evaluasi fungsi-fungsi ini dalam hal throughput, penggunaan memori, dan efek longsoran. Temuan menunjukkan bahwa meskipun SHAKE128 dan SHAKE256 menunjukkan throughput yang superior, namun keduanya membutuhkan memori yang lebih besar, terutama dengan ukuran input yang lebih besar. BLAKE2s menunjukkan keseimbangan yang kuat antara throughput, efisiensi memori, dan efek longsoran yang konsisten, sehingga menjadikannya opsi yang dapat diandalkan untuk output 256-bit. Sebaliknya, xxHash dan SipHash memberikan throughput yang tinggi dan penggunaan memori yang minimal, namun menunjukkan efek longsoran yang lebih rendah. Temuan penelitian ini memberikan wawasan penting bagi para pengembang dan peneliti dalam pemilihan solusi kriptografi yang tepat, yang harus disesuaikan dengan batasan dan persyaratan keamanan perangkat IoT.*

Kata Kunci*: analisis peformansi, efisiensi memori, fungsi hash, keamanan IoT, kriptografi ringan*

I. INTRODUCTION

The Internet of Things (IoT) across a myriad of sectors such as healthcare, automotive, smart homes, and agriculture marks a significant evolution in how technology integrates into daily operations and decision-making processes [1]–[3]. The concept of the Internet of Things (IoT) encompasses a network of interconnected devices, sensors, and systems that facilitate communication and data sharing, thereby enhancing operational efficiency, improving decision-making processes, and providing new services and functionalities [4]–[6]. A study conducted by Grand View Research indicates that the global IoT market in 2023 is valued at USD 1.18 billion and is projected to grow at a compound annual growth rate (CAGR) of 11.4% from 2024 to 2030, reflecting the widespread adoption and increasing reliance on this technology [7]. The capacity to gather, transmit, and analyze copious quantities of data in real-time represents a substantial benefit conferred by IoT systems, thereby facilitating the creation of more intelligent environments and the generation of more informed decisions [6], [8], [9].

Nevertheless, the incorporation of Internet of Things (IoT) devices into critical infrastructures has also given rise to novel challenges, particularly in regard to security. Internet of Things (IoT) devices frequently operate under considerable resource constraints, including limited processing power, minimal memory, and low energy capacities [10], [11]. These limitations present significant challenges in the implementation of traditional cryptographic

techniques, which are typically resource-intensive. Consequently, there is a compelling necessity for the development of lightweight cryptographic solutions that can guarantee the security and integrity of data while operating effectively within the constrained environments of IoT devices [12].

This paper aims to examine the effectiveness of various lightweight hash functions designed for Internet of Things (IoT) applications, with a particular focus on measuring computational overhead, memory usage, and security robustness. Functions such as SHAKE128, BLAKE2s, and SHAKE256 have been designed to address the unique challenges posed by the constrained resources of IoT devices, offering a balance between security and efficiency [13]–[15]. The objective of this study is to evaluate the efficacy of lightweight hash functions in addressing security challenges and resource limitations in IoT applications. It is hypothesized that lightweight hash functions specifically designed for IoT environments will exhibit advantages in resource efficiency while maintaining high security integrity compared to traditional hash functions. Furthermore, this research will investigate the avalanche effect of each hash function, which is a crucial element in evaluating the security of hashes. It is hypothesized that minor alterations to the input data will result in substantial alterations to the output, thereby enhancing data security against attempts to predict the hash value or cause collisions. This is a particularly crucial challenge in IoT networks, where data frequently traverses multiple nodes [16].

The Internet of Things (IoT) security landscape is rapidly evolving, facing new challenges as devices proliferate. Recent studies show that reported attacks on IoT infrastructure underscore the urgent need for a more effective and adaptive cryptographic response. This response must take into account the operational constraints of IoT devices and the evolving security threats. The choice of cryptographic primitives is particularly important in environments where the consequences of a security breach can be severe, such as healthcare or industrial control [17].

The integration of cryptographic hash functions in Internet of Things (IoT) systems has become an important aspect of data management, given the exponential growth in the number of connected devices that require secure and efficient data management. Cryptographic hash functions are essential to ensure the integrity and authenticity of data exchanged in the context of the IoT ecosystem, ensuring that any changes to the input data can be easily detected [18]. Cryptographic hash functions are designed to take an input or "message" and return a fixed-size string of bytes, typically a unique hash value for each unique input. They can be used to verify data integrity, generate digital signatures, and ensure secure communication between Internet of Things (IoT) devices [19]. The integrity of a hash function is critical in cryptographic applications. It must be deterministic, computationally efficient, irreversible, able to avoid collisions, and sensitive to changes in the input data, a phenomenon known as the avalanche effect [20].

In particular, the importance of lightweight hash functions has been extensively explored in the literature. However, these evaluations often only highlight the effectiveness of the solution without integrating the underlying weaknesses. Traditional hash functions such as SHA-256 and MD5 have been relied upon for their high level of security, but studies show that they are inefficient in resource-constrained environments such as the IoT due to their computational and energy intensive nature [13]. The lightweight alternatives that have been developed, such as LightMAC and CHASKEY, offer improved energy efficiency but are often poorly tested in complex attack scenarios, calling into question their robustness against serious threats [21]. The ASCON family, specifically ASCON-HASH, has been selected by NIST as the standard for lightweight cryptography. Despite its selection, research has identified potential vulnerabilities, such as impractical collision attacks on 2-round ASCON-HASH, which highlight the need for continued evaluation and improvement of this algorithm [22]. To address these challenges, RECO-ASCON, a reconfigurable security processor, has been developed to support ASCON hash functions with improved power efficiency and adaptability across multiple hardware platforms, making it suitable for various IoT systems [23]. In addition, the NIST standardization process has resulted in the optimization of several lightweight hash functions, including PHOTON-Beetle, Sparkle, Ascon, and Xoodyak. These implementations focus on achieving a balance between computation time and hardware area, with PHOTON-Beetle achieving a very small hardware footprint and Ascon and Xoodyak offering a high throughput-to-area ratio [24].

This analysis attempts to integrate performance benchmarks with security assessments to propose optimizations that improve not only the utility but also the robustness of lightweight hash functions in the context of IoT. The research, which aims to provide forward-looking guidance for designing effective hash functions in constrained environments, also highlights the importance of verifying data integrity and authenticity. Hash functions in IoT should be deterministic, computationally efficient, irreversible, able to avoid collisions, and sensitive to changes in the input data. However, the main challenge is to balance security and efficiency without sacrificing one for the other.

Recent developments show that despite significant progress in the optimization of lightweight cryptographic algorithms, there is still a need for further research to improve not only security but also efficiency in IoT applications. Therefore, the selection of hash functions should be done with critical consideration of their ability to address real threats in an IoT environment, not just their theoretical performance under laboratory conditions [25].

This research is important because it fills an existing gap in the literature related to the use of lightweight hash functions in highly dynamic and diverse IoT environments. Although previous research has identified several algorithms that meet the specific needs of the IoT in terms of efficiency and security, many of these studies do not fully address potential weaknesses in real-world implementations or specific operational requirements. This research not only evaluates the efficiency and security of lightweight hash functions, but also tests their robustness against

attacks in real-world scenarios, providing deeper and more applicable insights for the development of secure IoT systems.

Furthermore, the results of this research will provide evidence-based guidelines for developers and security engineers to select or design the most appropriate hash functions for their IoT applications, given resource constraints and potential security risks. This approach ensures data integrity and security, two critical aspects that cannot be ignored in the development of IoT technologies, especially in sensitive sectors such as healthcare and industry.

Theoretically, this research advocates for the revision of standards and best practices in IoT data security by suggesting the implementation of more stringent and adaptive security principles. Practically, the findings are expected to help improve the resilience of IoT infrastructure against increasingly sophisticated attacks, facilitating the development of IoT devices that are not only efficient, but also highly secure, supporting the continued expansion and adoption of IoT technology in various sectors.

II. METHODOLOGY

The research methodology for analyzing the performance and security of lightweight hash functions in IoT devices employs a combination of experimental and theoretical approaches to ensure comprehensive coverage of technical efficacy and real-world applicability. The methodology is structured into several key phases, each of which is designed to address specific aspects of the research question. This approach ensures that the results are both robust and relevant to the unique constraints of IoT environments.

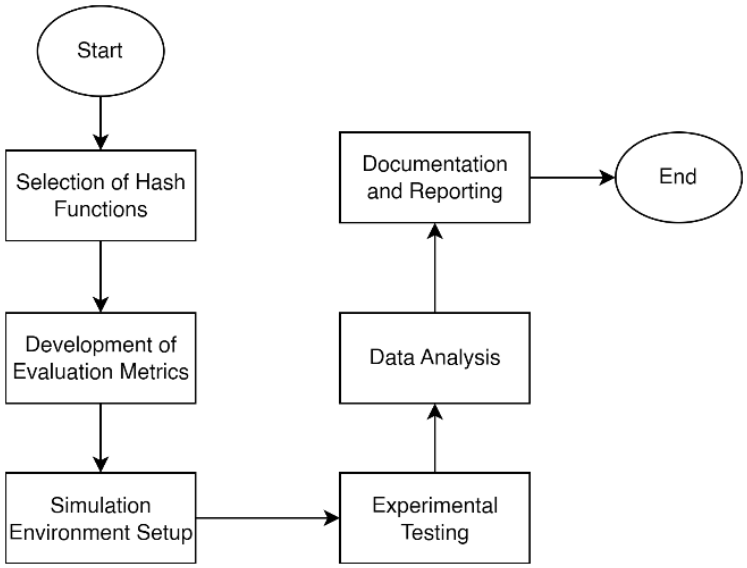

Figure 1 Research methodology

The initial phase of the process entails the meticulous selection of lightweight hash functions to be evaluated. The hash functions selected for this study—SHA-256, SHAKE128, BLAKE2s, SHA3-256, SipHash, and xxHash—have been demonstrated to have minimal computational requirements and robust security features, making them suitable candidates for IoT applications. The selection process is based on several criteria, including the hash function's prevalence in existing literature, its demonstrated security properties, and its efficacy in resource-constrained environments. This phase guarantees that the hash functions tested encompass a diverse range of cryptographic approaches, thereby facilitating a comprehensive comparison across different metrics.

The experimental setup is designed to mimic the operating conditions typically faced by IoT devices, with a particular focus on the performance and security characteristics of the chosen hash function. This research utilizes a dual simulation environment, consisting of software simulations performed using the Arduino Integrated Development Environment (IDE), and physical testing on a dedicated hardware platform. The Arduino Uno, a microcontroller commonly used for prototyping IoT devices, was used to simulate the resource-constrained environment typical of such devices. This configuration allows the practical performance of each hash function to be evaluated under conditions similar to actual IoT deployments.

In order to evaluate the selected hash functions, the study develops a set of metrics that cover both performance and security aspects. The performance metrics include computational time, memory usage, and energy consumption, which are critical factors in IoT environments where resources are limited. Computational time quantifies the speed

with which a hash function processes data, memory usage assesses the amount of memory required to execute the hash function, and energy consumption quantifies the power required to perform the hashing operation.

With regard to the security of the selected hash functions, the study considers their resilience to common cryptographic attacks, including collision, pre-image, and second pre-image attacks. Furthermore, the study assesses the avalanche effect of each hash function, which is the extent to which minor alterations in the input result in substantial modifications to the output. A robust avalanche effect is an indicator of a secure hash function, as it ensures that any minor alteration in the input results in a markedly different hash output, thereby enhancing security.

Data from the experiments was analyzed using specialized statistical software to identify patterns and anomalies in performance and security metrics. The analysis used a combination of descriptive and inferential statistics to infer performance differences between hash functions. In addition, the analysis includes a comparison of hash function performance against theoretical expectations to identify any deviations between predicted and actual performance. These comparisons are important for elucidating the extent to which these hash functions can be translated from theory to practice, particularly in the context of resource-constrained IoT devices.

Moreover, the analysis includes a comparison of the hash functions' performance against theoretical expectations, thus enabling the identification of any discrepancies between predicted and actual performance. This comparison is of particular significance for elucidating the extent to which these hash functions are able to be translated from theory to practice, particularly in the context of resource-constrained Internet of Things (IoT) devices.

The final phase of the research involves a detailed documentation of the methodologies, tests, findings, and conclusions. This comprehensive report not only provides insights into the performance and security of lightweight hash functions in IoT devices but also offers practical recommendations for developers and researchers in the field. The outcome of this study is expected to contribute significantly to the advancement of cryptographic practices within the IoT ecosystem, ensuring that security implementations are both effective and practical for deployment in resource-constrained environments.

III. RESULT AND DISCUSSION

Experiments were conducted on several hash functions on the Arduino Uno microcontroller with the same data for each hash function. Each experiment used different input data lengths. Three parameters were measured in this experiment: throughput, memory usage, and avalanche effect. The results of the experiments are as follows:

A. Throughput

This is the result of throughput experimental between various lightweight hash functions shown in TABLE 1.

TABLE 1

THROUGHPUT MEASUREMENT

Hash Function	Input Data Size (bit)							
	32	***64***	***128***	***256***	***512***	***1024***	***2048***	***4096***
SHAKE128	480,54	961,54	1921,23	3833,25	7637,23	15151,51	15245,36	15287,23
BLAKE2s	1122,33	2244,67	4489,34	8978,68	17917,13	18068,89	18125,18	18150,88
SHA256	369,41	738,83	1477,1	2954,21	2965,16	3956,48	4697,16	5641,33
SHA3 256	480,54	961,08	1919,39	3829,58	7626,31	15115,73	15205,51	15258,08
SipHash	4000,00	8000,00	8000,00	2133,33	3368,42	7111,11	14222,22	24380,95
xxHash	37037,03	74074,07	148148,14	83333,33	109589,04	128514,06	141289,35	147976,87

The analysis concludes that SHAKE128 and SHAKE256 have consistent and stable throughput across all input sizes, making them highly efficient for various data sizes. BLAKE2s shows excellent throughput, especially for large data sizes, making it highly efficient for 256-bit output. Meanwhile, SHA-256 has lower throughput compared to BLAKE2s and SHA3-256 but remains stable. In terms of best performance, xxHash shows very high throughput across all input sizes, making it the most efficient hash function in this test. SipHash also shows excellent performance for small block sizes with significant throughput for large data sizes. For applications requiring high efficiency for large data sizes, xxHash and SHAKE128 are the best choices. For general needs with 256-bit output, BLAKE2s and SHA3-256 are reliable. SipHash is well-suited for applications with small block sizes that require high performance. Throughput measurements of the hash functions show significant performance variations. SHAKE128 and SHAKE256 have consistent and stable throughput across all input sizes, making them highly efficient for various data sizes. BLAKE2s shows excellent throughput, especially for large data sizes, making it a highly efficient choice for 256-bit output. SHA-256 has lower throughput compared to BLAKE2s and

SHA3-256 but remains stable. xxHash shows the best performance with very high throughput across all input sizes, making it the most efficient hash function in this test. SipHash also shows excellent performance for small block sizes with significant throughput for large data sizes.

B. Memory Usage

This is the result of memory usage experimental between various lightweight hash functions shown in TABLE 2.

TABLE 2

MEMORY USAGE MEASUREMENT

Hash Function	Input Data Size (bit)							
	32	***64***	***128***	***256***	***512***	***1024***	***2048***	***4096***
SHAKE128	43%	43%	44%	45%	46%	49%	56%	68%
BLAKE2s	24%	24%	25%	25%	27%	30%	36%	49%
SHA256	24%	24%	25%	25%	27%	30%	36%	49%
SHA3 256	29%	29%	29%	30%	32%	35%	41%	54%
SipHash	11%	11%	12%	12%	14%	17%	23%	36%
xxHash	13%	13%	13%	14%	16%	19%	25%	37%

The experiment results on hash function memory usage on Arduino Uno show several important points. SHAKE128 shows a significant increase in memory usage with increasing input size, reaching up to 68%. This indicates that SHAKE128 requires more memory for larger data. BLAKE2s and SHA-256 have the same relatively stable and efficient memory usage even for larger input sizes, with a peak usage of 49%. SHA3-256 uses more memory compared to BLAKE2s and SHA-256 but still within acceptable limits, reaching 54%.

In terms of best performance for small block sizes, SipHash and xxHash show very low and stable memory usage even for the largest input sizes, reaching 36% and 37%, respectively. This makes them very efficient and suitable for applications with memory constraints.

For applications requiring memory efficiency across various input sizes, SipHash and xxHash are the best choices due to their low and stable memory usage. BLAKE2s and SHA-256 are also recommended for applications needing 256-bit output with efficient memory usage. SHAKE128 and SHA3-256 can be used for applications that can accommodate increased memory usage, especially for larger input sizes.

Memory usage is a crucial factor, especially on platforms with limitations like Arduino Uno. SHAKE128 shows a significant increase in memory usage with increasing input size, reaching up to 68%. BLAKE2s and SHA-256 have relatively stable and efficient memory usage even for larger input sizes, with peak usage at 49%. SHA3-256 uses more memory compared to BLAKE2s and SHA-256 but still within acceptable limits, reaching 54%. For small block sizes, SipHash and xxHash show very low and stable memory usage even for the largest input sizes, reaching 36% and 37%, respectively.

.

C. Avalanche Effect

This is the result of avalanche effect experimental between various lightweight hash functions shown in TABLE 3.

TABLE 3

AVALANCHE EFFECT MEASUREMENT

Hash Function	Input Data Size (bit)							
	32	*64*	*128*	*256*	*512*	*1024*	*2048*	*4096*
SHAKE128	52,34%	50,00%	49,22%	49,22%	57,03%	46,09%	-	-
BLAKE2s	48,05%	48,05%	48,83%	53,91%	47,27%	53,13%	51,95%	48,83%
SHA256	50,00%	48,44%	47,27%	53,52%	53,52%	50,00%	47,66%	49,61%
SHA3 256	48,83%	46,88%	45,70%	43,75%	50,78%	51,95%	43,36%	-
SipHash	39,06%	51,56%	50,00%	53,13%	51,56%	65,62%	51,56%	53,13%
xxHash	34,38%	31,25%	21,87%	25,00%	43,75%	21,87%	37,50%	15,63%

SHAKE128 shows inconsistent avalanche effect variations across different input sizes, with the lowest at 1024 bits and highest at 512 bits, reaching up to 5703%. BLAKE2s has stable and consistent avalanche effects across various input sizes, ranging from 4805% to 5391%, showing reliable characteristics in distributing output bits that change with small changes in input. SHA-256 shows relatively stable performance with avalanche effects ranging from 4477% to 5352%, while SHA3-256 shows good stability with some fluctuations ranging from 4375% to 5195%.

For small block sizes, SipHash shows significant variation in avalanche effects, ranging from 3906% to 6562%, with the highest value at 1024-bit input size. xxHash shows lower avalanche effect values compared to other hash functions, ranging from 1563% to 4375%, indicating less effective performance in generating random bit distributions.

Based on this analysis, BLAKE2s and SHA-256 can be recommended for applications requiring consistent and reliable output bit distributions. SipHash can be used for applications needing good performance for small block sizes, despite larger variations. SHAKE128 and SHA3-256 can also be used, but variations in their avalanche effects should be noted. xxHash may be less ideal for applications requiring high avalanche effect levels.

Avalanche effect measures how well small changes in input result in significant changes in output. SHAKE128 shows inconsistent variations in avalanche effects across different input sizes, with the highest value at 512 bits and the lowest at 1024 bits. BLAKE2s has stable and consistent avalanche effects across various input sizes, ranging from 4805% to 5391%, showing reliable characteristics. SHA-256 and SHA3-256 also show good stability with some fluctuations within acceptable ranges. For small block sizes, SipHash shows significant variations with the highest value at 1024-bit input size, while xxHash shows lower avalanche effect values compared to other hash functions.

IV. CONCLUSION

A comprehensive analysis of lightweight hash functions on Arduino Uno reveals the strengths and limitations of different algorithms in terms of throughput, memory consumption, and avalanche effect. SHAKE128 and SHAKE256 are characterized by high throughput efficiency, but require more memory for larger input sizes and show variations in avalanche effects. For the combination of high throughput, efficient memory usage, and stable avalanche effect, BLAKE2s proved to be a reliable choice, especially for 256-bit output.

Meanwhile, SHA-256 and SHA3-256 provide efficient memory usage with a stable avalanche effect, although their throughput is lower compared to BLAKE2s. For smaller block sizes, SipHash shows excellent performance in terms of throughput and low memory consumption, albeit with variations in avalanche effect. xxHash, with its very high throughput and low memory consumption, is ideal for applications that require high efficiency for large data sizes, despite its lower avalanche effect.

Recommendations for hash function selection should take into account specific application needs, especially input data size, memory efficiency, and avalanche stability to ensure optimal performance on platforms such as the Arduino Uno. For general needs with 256-bit output, BLAKE2s and SHA-256 are highly recommended due to their memory efficiency and stable avalanche effect. SHAKE128, although its avalanche effect variation requires scrutiny, and SHA3-256 are suitable for applications that require a balance between throughput, memory consumption, and avalanche effect.

Given these results, the proper selection of hash functions can significantly improve the security and efficiency of IoT applications. It is recommended that IoT hardware and software developers consider using SHAKE128 and BLAKE2s for applications that require high throughput and have adequate memory availability. Further research

should be directed at developing optimization techniques that can reduce the memory footprint of SHAKE128 without sacrificing performance, as well as investigating new algorithms that produce more stable avalanche effects. Extensive testing on real-world applications is needed to evaluate the performance and security of hash functions such as SHA3-256 and xxHash under various operating conditions, and researchers are expected to develop a standardized testing framework for more consistent evaluation of the avalanche effect. Developers in the healthcare and industrial sectors - where reliability and security are critical - should use BLAKE2s and SHA-256 for storage efficiency and stability. Research and publication of these findings in relevant industry forums and journals will go a long way toward advancing the practical application of these technologies in the field.

ACKNOWLEDGMENT

The financial support of the Indonesia's DRTPM, DITJEN DIKTIRISTEK, KEMDIKBUDRISTEK through grant 106/E5/PG.02.00.PL/2024, 043/SP2H/RT-MONO/LL4/2024 and 078/LIT07/PPM-LIT/2024 is hereby acknowledged and appreciated.

REFERENCES

[1] I. Adhicandra, T. Tanwir, A. Asfahani, J. W. Sitopu, and F. Irawan, "Latest Innovations in Internet of Things (IoT): Digital Transformation Across Industries," *Innov. J. Soc. Sci. Res.*, vol. 4, no. 3, pp. 1027–1037, 2024, doi: 10.31004/innovative.v4i3.10551.

[2] S. Dhar, A. Khare, A. D. Dwivedi, and R. Singh, "Securing IoT devices: A novel approach using blockchain and quantum cryptography," *Internet of Things*, vol. 25, p. 101019, 2024, doi: https://doi.org/10.1016/j.iot.2023.101019.

[3] Z. Khudoykulov, "A Comparison of Lightweight Cryptographic Algorithms," Springer International Publishing, 2024, pp. 295–304. doi: 10.1007/978-3-031-53488-1_36.

[4] Z. Kang, A. Hamid, K. Ahmad, and N. Ali, "Research on internet of things (IoT) indexed in Web of Science from 2011–2022," *Inf. Dev.*, vol. 0, no. 0, p. 02666669241247783, doi: 10.1177/02666669241247783.

[5] H. D. Doss and A. Mishra, "The Internet of Things (IoT)," 2024. doi: 10.1201/9781003474838-1.

[6] R. Mahajan, R. Arora, R. Mahajan, and R. Arora, "Data Analysis Using IoT Technologies for Enhanced Healthcare Decision-Making," in *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/979-8-3693-2909-2.ch008*, IGI Global, 2024, pp. 100–110. doi: 10.4018/979-8-3693-2909-2.ch008.

[7] "Internet of Things (IoT) Market Size, Share & Trends Analysis Report by Component (Hardware, Software, Services), By Deployment (On-premise, Cloud), By Connectivity, By End-use, By Region, And Segment Forecasts, 2024 - 2030," 2024. https://www.grandviewresearch.com/industry-analysis/iot-market (accessed Oct. 07, 2024).

[8] S. Malik, "Data-Driven Decision-Making : Leveraging the IoT for Real-Time Sustainability in Organizational Behavior," 2024.

[9] M. F. Zahra, "Manajemen Data Real-Time Untuk Aplikasi Internet Of Things (IOT)," no. 2, 2024.

[10] K. Singh *et al.*, "Finding Security Gaps and Vulnerabilities in IoT Devices," in *https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/979-8-3693-6016-3.ch023*, IGI Global, 2024, pp. 379–395. doi: 10.4018/979-8-3693-6016-3.ch023.

[11] S. Dhar, A. Khare, A. D. Dwivedi, and R. Singh, "Securing IoT devices: A novel approach using blockchain and quantum cryptography," *Internet of things*, 2023, doi: 10.1016/j.iot.2023.101019.

[12] S. S. Dhanda, B. Singh, and P. Jindal, "Lightweight Cryptography: A Solution to Secure IoT," *Wirel. Pers. Commun.*, vol. 112, no. 3, pp. 1947–1980, 2020, doi: 10.1007/s11277-020-07134-3.

[13] S. D. E. V Achar and J. S. S. Academy, "LiteHash : Hash Functions for Resource-Constrained Hardware," 2024, doi: 10.1145/3677181.

[14] S. Windarta, S. Suryadi, K. Ramli, B. Pranggono, and T. S. Gunawan, "Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions," *IEEE Access*, vol. 10, pp. 82272–82294, 2022, doi: 10.1109/ACCESS.2022.3195572.

[15] S. Windarta *et al.*, "Two New Lightweight Cryptographic Hash Functions Based on Saturnin and Beetle for the Internet of Things," *IEEE Access*, vol. 11, no. July, pp. 84074–84090, 2023, doi: 10.1109/ACCESS.2023.3301128.

[16] D. Upadhyay, N. Gaikwad, M. Zaman, and S. Sampalli, "Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications," *IEEE Access*, vol. 10, pp. 112472–112486, 2022, doi: 10.1109/ACCESS.2022.3215778.

[17] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Networks*, vol. 57, no. 10, pp. 2266–2279, 2013, doi: https://doi.org/10.1016/j.comnet.2012.12.018.

[18] "Challenges and Opportunities in Maintaining Data Integrity Shah Rukh Khan Department of Computer Science , University of Cambridge," pp. 1–7.

[19] Y. N.-E. Aine and C. Leghris, "Securing IoT Communication: A Steganographic Protocol for Efficient Mutual Authentication and Data Integrity," in *2024 IEEE 12th International Symposium on Signal, Image, Video and Communications (ISIVC)*, 2024, pp. 1–6. doi: 10.1109/ISIVC61350.2024.10577843.

[20] G. Bertoni, J. Daemen, and G. Van Assche, "The Keccak SHA-3 submission," pp. 1–14, 2011.

[21] B. Kaur, M. Rakhra, D. Singh, A. Singh, and Shruti, "Advancements in Lightweight Cryptography: Secure Solutions for Resource-Constrained Environments in IoT, WSNs, and CPS," in *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2024, pp. 1–7. doi: 10.1109/ICRITO61523.2024.10522297.

[22] D. Zhai, W. Bai, J. Fu, H. Gao, and X. Zhu, "Improved 2-round collision attack on IoT hash standard ASCON-HASH," *Heliyon*, vol. 10, p. e26119, 2024, doi: 10.1016/j.heliyon.2024.e26119.

[23] M. El-Hadedy *et al.*, "RECO-ASCON: Reconfigurable ASCON hash functions for IoT applications," *Integration*, vol. 93, p. 102061, 2023, doi: 10.1016/j.vlsi.2023.102061.

[24] S. Khan, W. K. Lee, A. Karmakar, J. Mera, A. Majeed, and S. Hwang, "Area–Time Efficient Implementation of NIST Lightweight Hash Functions Targeting IoT Applications," *IEEE Internet Things J.*, vol. PP, p. 1, 2022, doi: 10.1109/JIOT.2022.3229516.

[25] M. Stevens, "Attacks on Hash Functions and Applications," *Univ. Leiden*, p. 258, 2012, [Online]. Available: http://marc-stevens.nl/research/papers/PhD Thesis Marc Stevens - Attacks on Hash Functions and Applications.pdf