ISSN 2347 - 3983
Volume 12. No.8, August 2024
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter011282024.pdf
https://doi.org/10.30534/ijeter/2024/011282024

A QoS Analysis of SHA Algorithms for IoT Systems

Zied Guitouni[1], Eya Ben Brahim[2], Mounir Zrigui[2], Mohsen Machhout[1]

[1] Electronics and Microelectronics Laboratory, Faculty of Sciences Monastir (FSM), Tunisia
[2] Research Laboratory in Algebra, Numbers theory and Intelligent Systems, FSM, Tunisia
guitounizied@yahoo.frl

Received Date: June 25, 2024 Accepted Date: July 29, 2024 Published Date : August 07, 2024

ABSTRACT

Secure Hash Algorithms (SHA) are widely used in the Internet of Things (IoT) systems for message authentication and integrity verification. However, the performance of different SHA algorithms can vary significantly in terms of Quality of Service (QoS) metrics such as area utilization, processing speed, energy efficiency, and security. In this paper, we present a comprehensive analysis of the QoS parameters of various SHA algorithms and discuss the trade-offs between performance and security when selecting SHA algorithms for resource-constrained IoT devices. The study focuses on the hardware implementation of SHA algorithms in Field-Programmable Gate Array (FPGA) devices, which are commonly used in IoT applications. The performances and resource utilization of different SHA algorithms are compared and analyzed. The comparative results show that SHA-2 provide a good balance between performance and security, but SHA-3 provide better security due to its resistance to attacks such as length extension and collision

Key words : Secure Hash Algorithms, IoT systems, Quality of Service, QoS parameters, resource-constrained devices, FPGA

1. INTRODUCTION

The Internet of Things (IoT) has become an essential part of modern life, with billions of devices interconnected through wireless networks. As the use of IoT systems increases, so do the security concerns associated with them. Cryptographic algorithms, such as SHA algorithms, are widely used to secure the transmission of data in IoT systems. However, the performance of SHA algorithms is an essential consideration when dealing with resource-limited IoT devices.

Quality of Service (QoS) is a critical factor in designing IoT systems that meet the performance requirements of users. In particular, the performance of cryptographic algorithms must be optimized to ensure that IoT devices operate efficiently while maintaining the security and privacy of transmitted data. One important factor that affects the QoS of hash functions is their computational complexity. Hash functions with higher computational complexity may require more processing power and memory, which can affect the performance of resource-constrained IoT devices. Therefore, the selection of hash functions with an appropriate level of computational complexity is critical to ensure the desired level of QoS in IoT systems.

In this paper, we provide a QoS analysis of SHA algorithms in IoT systems. We investigate the performance of three different SHA algorithms, namely SHA-1, SHA-2, and SHA-3, in terms of their resource utilization, processing speed, power consumption, and security analysis. Our study aims to provide insights into the performance of SHA algorithms and to identify which algorithm is best suited for IoT systems. We also discuss the trade-offs between security, performance, and QoS when selecting an appropriate SHA algorithm for IoT devices. The rest of the paper is organized as follows. Section 2 provides a brief overview of SHA algorithms and their use in IoT systems. Section 3 details the different SHA standards recommended by NIST including SHA-1, SHA-2, and SHA-3. The techniques and results of FPGA implementation of different SHA algorithms are described in Section 4. Section 5 discusses a security analysis of SHA algorithms. Finally, Section 6 concludes the paper.

2. IoT SYSTEMS SECURITY USING SHA FMILLY

2.1 IoT Systems Security Description

IoT systems are highly vulnerable to security threats due to the massive amount of data transmitted across various networks [1]. Ensuring the security of IoT systems is critical to safeguard sensitive information from unauthorized access, tampering, or theft. SHA algorithms provide an effective solution for securing IoT systems by offering various security features.

Data integrity is one of the primary use cases of SHA algorithms in IoT systems. SHA algorithms generate unique message digests that act as a fingerprint for data packets [2]. These message digests are used to verify the integrity of the data during transmission. The receiving device calculates the message digest of the received data and compares it with the expected message digest. If the two digests match, it ensures that the data has not been tampered with during transmission.

Another significant use case of SHA algorithms in IoT systems is message authentication. In combination with digital signatures or message authentication codes (MACs), SHA algorithms provide message authentication capabilities [3]. The sender generates a

hash or HMAC of the message using a private key, and the recipient verifies the authenticity by computing the hash using the corresponding public key. This ensures that the message originated from a trusted source and was not modified during transit [4].

SHA algorithms are also used in firmware and software verification in IoT devices. IoT devices require periodic firmware and software updates to fix security vulnerabilities and add new features [5]. SHA algorithms are used to verify the authenticity and integrity of these updates. The device generates a message digest of the firmware or software package using SHA algorithms during the update process. The receiving party can then compare the received message digest with the calculated one to ensure the authenticity and integrity of the update.

Another use case of SHA algorithms in IoT systems is secure boot. Secure boot ensures that only authenticated and trusted software runs on IoT devices. SHA algorithms generate message digests of firmware and software components. The device compares the received message digest with the calculated one during boot-up, and if they match, the firmware and software are loaded, ensuring that only authorized and tamper-free software runs on the device.

2.2 Related Works

In this section, a brief description of each related works on IoT systems security using SHA algorithms: Alghamdi et al [6] present a comprehensive survey of SHA algorithms and their variants, including their security features and applications. The authors also discuss the challenges and research directions of SHA algorithms, which can be helpful for researchers and practitioners in the field of IoT security. A survey of security challenges and solutions in IoT-based Healthcare Systems, was described by Afzal et al [7], this work proposes a secure communication framework using the SHA-256 algorithm to protect the sensitive data transmitted in healthcare systems. In [8] Singh et al. propose a secure communication framework for IoT systems using the SHA-512 algorithm, which can protect data integrity and confidentiality. The authors also evaluate the performance of the proposed framework in terms of execution time and security. Gao et al [9] investigate the application of the SHA-3 algorithm in IoT security and propose a secure communication framework that can protect data confidentiality and integrity. The authors also evaluate the performance of the proposed framework in terms of security and efficiency. In [10] Zhang et al propose a secure and efficient communication protocol for IoT systems using SHA-256 and ECC (Elliptic Curve Cryptography) algorithms. The authors also evaluate the performance of the proposed protocol in terms of security and efficiency. Bao et al [11] described a lightweight secure communication protocol for IoT systems using SHA-256 and ChaCha20 algorithms. The authors also evaluate the performance of the proposed protocol in terms of security and efficiency. Wang et al [12] propose a secure and efficient communication protocol for IoT systems using SHA-256 and AES (Advanced Encryption Standard) algorithms and evaluate the performance of the proposed protocol in terms of security and efficiency. In [13] Liu et al propose a secure and efficient communication protocol for IoT systems using SHA-256 and SM4 algorithms. Vasilakos et al [14] described security challenges and opportunities in the industrial Internet of Things. This work reviews the security challenges and opportunities in industrial IoT systems and proposes a secure communication framework using the SHA-256 algorithm to protect the sensitive data transmitted in the system. Wang et al [15] propose a secure and efficient communication protocol for IoT systems using SHA-256 and SM2 J. Raja et al [16], proposes the use of the SHA3 algorithm for secure data transmission in IoT systems. The authors demonstrate the effectiveness of the proposed approach through simulations.

3. SHA ALGORITHMS

In this section, a brief description of the standard SHA algorithms is described.

3.1 SHA-1 Algorithm

SHA-1 is a cryptographic hash function that produces a 160-bit hash value. It takes an input message of arbitrary length and produces a fixed-size output, which is a unique digital fingerprint of the message. The SHA-1 algorithm is designed to be a one-way function, meaning it is practically impossible to determine the original input message from the hash value.

The algorithm consists of four main stages: message padding, message processing, hash value initialization, and hash value output. In the message padding stage, the input message is padded with a bit sequence to ensure that its length is a multiple of 512 bits. The padded message is then divided into 512-bit blocks and processed in the message processing stage.

In the message processing stage, each block is processed using a compression function that combines the current block with the previous hash value. This function involves a series of logical operations, including bitwise operations, modular arithmetic, and message expansion.

In the hash value initialization stage, a fixed initial hash value is loaded into the algorithm. This value is used as the starting point for the compression function in the message processing stage. In the hash value output stage, the final 160-bit hash value is produced by concatenating the output of the compression function for each block of the message.

SHA-1 consists of four rounds of operations, with each round performing a series of logical and bitwise operations on the input data. Here's a high-level description of the four rounds:

- ***Round 1***: In the first round of SHA-1, the input data is divided into 16 words, denoted as W0, W1, ..., W15. Each word consists of 32 bits. These words are individually processed through a series of logical functions, including Boolean operations such as bitwise AND, OR, XOR, and NOT. Additionally, a constant value specific to the round, denoted as K0, is combined with the result of the logical functions. This constant value enhances the non-linearity and cryptographic properties of the algorithm.
- ***Round 2***: Similar to Round 1, Round 2 also divides the input data into 16 words, denoted as W16, W17,..., W31. Each word undergoes a different set of logical functions compared to the previous round. Furthermore, a distinct constant value, denoted as K1, is combined with the output of the logical functions. The utilization of different constants for each round ensures that each

round introduces additional complexity and non-linearity into the hash computation.

• ***Round 3***: In Round 3, the input data is once again divided into 16 words, denoted as W32, W33, ..., W47. These words are subjected to a unique set of logical functions, distinct from the operations performed in the previous two rounds. Similarly, a specific constant value, denoted as K2, is combined with the output of the logical functions. The inclusion of distinct constants in each round further enhances the diffusion and confusion properties of the hash function.

• ***Round 4***: The final round of SHA-1 involves dividing the input data into 20 words, denoted as W48, W49, ..., W67. Each word undergoes a unique set of logical operations, and a specific constant value, denoted as K3, is combined with the output of the logical functions. This round serves as the last step in the hash computation process.

Figure 1, shows the description of the SHA-1 rounds.

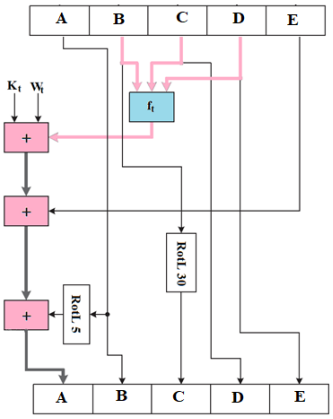

Figure 1: SHA-1 Rounds description

3.2 SHA-2 Algorithm

SHA-2 is a family of cryptographic hash functions that includes SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. It is an improvement over the SHA-1 algorithm, providing stronger security and longer hash output sizes. The SHA-2 family is widely used in security applications, such as digital signatures, message authentication codes, and password storage.

The SHA-2 algorithm works by taking an input message of arbitrary length and processing it in blocks of 512 bits. It then applies a series of logical operations, including bitwise operations, modular arithmetic, and message expansion, to generate a fixed-size output, which is a unique digital fingerprint of the message.

The SHA-2 family consists of different hash functions that have varying output sizes and security levels. SHA-224 produces a 224-bit hash value, SHA-256 produces a 256-bit hash value, SHA-384 produces a 384-bit hash value, and SHA-512 produces a 512-bit hash value. SHA-512/224 and SHA-512/256 produce truncated 224-bit and 256-bit hash values, respectively.

The SHA-2 algorithm uses a similar four-stage process as SHA-1, including message padding, message processing, hash value initialization, and hash value output. However, it involves a different compression function that incorporates more rounds of operations and is less susceptible to the vulnerabilities that affect SHA-1.

Figure 2 illustrates the computational structure of each round of SHA-2.

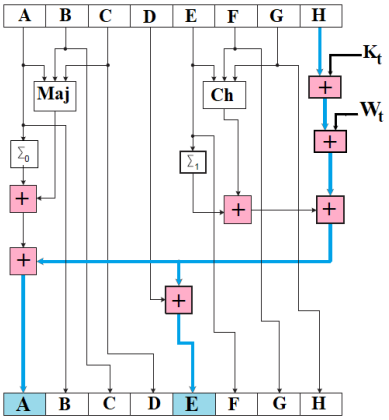

Figure 2: SHA-2 Rounds description

• The Majority (**Maj**) operation is a logical function performed within the SHA-2 rounds. It takes three input values, denoted as A, B, and C, and produces an output based on the majority of the three inputs. The Maj operation can be defined as follows:

$$Maj(A, B, C) = (A\ AND\ B)\ XOR\ (A\ AND\ C)\ XOR\ (B\ AND\ C) \quad (1)$$

The Maj operation ensures that the resulting output bit is set to 1 if the majority of the input bits are 1. Otherwise, the output bit is set to 0. By incorporating the Maj operation, SHA-2 achieves non-linearity and ensures that the output is highly sensitive to changes in the input values.

• The Choice (**Ch**) operation is another crucial operation performed within the SHA-2 rounds. It takes three input values, denoted as E, F, and G, and produces an output based on specific conditions. The Ch operation can be defined as follows:

$$Ch(E, F, G) = (E\ AND\ F)\ XOR\ (NOT\ E\ AND\ G) \quad (2)$$

The Ch operation selects bits from the input based on the value of E. If E is 1, the output bit is set to the XOR of F and G. If E is 0, the output bit is set to the logical complement of E and the value of G. The Ch operation contributes to the diffusion property of SHA-2 by ensuring that the output is influenced by all input bits.

3.3 SHA-3 Algorithm

SHA-3 is a cryptographic hash function that was designed by the National Institute of Standards and Technology (NIST) as a successor to SHA-2. It provides stronger security and faster processing speeds compared to SHA-2, making it a more suitable choice for a wide range of security applications. The SHA-3 family includes four hash functions, including SHA3-224, SHA3-256, SHA3-384, and SHA3-512, which produce hash values of varying sizes. Keccak algorithm was proposed as a candidate for the SHA-3 competition organized by NIST. It was eventually selected as the winner and became the basis for the SHA-3 standard.

Keccak works by taking an input message of arbitrary length and processing it in blocks of 1600 bits. It then applies a series of rounds of operations, including bitwise operations, modular

arithmetic, and permutation functions, to generate a fixed-size output, which is a unique digital fingerprint of the message. The construction of the sponge construction of this keccak algorithm is depicted in Figure 3.

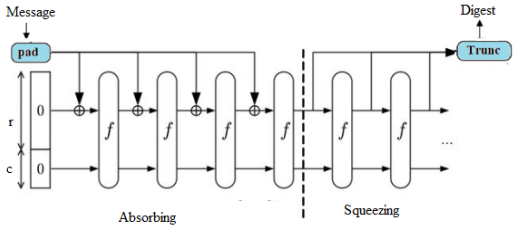

Figure 3: Sponge Construction

The sponge construction works by first dividing the input message into fixed-size blocks and then processing these blocks through a function that includes a permutation and a function that combines the input and output of the permutation. This is done repeatedly, with the output of the function being passed back as input to the permutation, until all the message blocks have been processed.

At this point, the output of the function is truncated to produce the desired hash value length. This allows for the creation of hash functions with variable-length outputs, as the length of the output can be chosen by selecting the number of bits to truncate from the final output.

3.4 Comparative Analysis of SHA Parameters

In Table 1, the comparative analysis of different SHA parameters is described.

Table 1: SHA Algorithms Parameters

SHA Algorithms		Hash Size	Block Size	Internal State	Rounds	Max Message
SHA-1		160	512	160	80	2^{64-1}
SHA-2	SHA224	224	512	256 (8 *32)	64	2^{64-1}
	SHA256	256				
	SHA384	384	1024	512 (6 * 64)	80	2^{128-1}
	SHA512	512				
SHA-3	224	160	1152	1024 (5 * 5* 64)	24	∞
	256	224	1088			
	384	256	832			
	512	384	576			

In the table 1, we have compared the digest size (Bits), the block size (Bits), the internal state size (Bits), the rounds number, and the Max Message Size of the different SHA algorithms

4. FPGA IMPLEMENTATION OF SHA ALGORITHMS

The FPGA implementation of SHA algorithms for IoT applications offers an efficient and flexible solution for securing IoT systems. FPGA-based implementations can provide high throughput and low power consumption while maintaining a high level of security, making them a suitable choice for various IoT applications. This section provides an overview of the FPGA implementation techniques for SHA algorithms and compares their resource utilization and speed on FPGA devices.

4.1 FPGA Implementation Techniques for SHA

The implementation of SHA algorithms in FPGA for IoT applications involves selecting an appropriate implementation technique that can provide high-speed cryptographic processing with low power consumption while maintaining a high level of security. The choice of implementation technique depends on the specific requirements of the IoT application, including desired throughput, area utilization, and frequency. In this section, we provide an overview of the various implementation techniques used in FPGA implementations of SHA algorithms for IoT applications.

- **Carry Select Adder (CSA):** Carry Select Adder (CSA) is a fast and efficient adder that uses multiple ripple carry adders with a select signal to choose the output of the correct adder. This technique can reduce the delay and power consumption of the adder by avoiding long carry chains [40].
- **Unfold:** Unfold is a technique that involves unfolding the loop of an iterative algorithm into multiple iterations to reduce the critical path delay. This technique can improve the performance of the algorithm by reducing the number of iterations required [42].
- **Parallel Prefix Logic (PPL):** Parallel Prefix Logic (PPL) is a technique that uses a network of logic gates to compute the prefix sum of a sequence of values. It can be used to compute the hash value of a message in parallel, reducing the computation time and improving the throughput [41].
- **Five-Parallel Prefix Logic (5PPL):** Five-Parallel Prefix Logic (5PPL) is a modification of the PPL technique that uses a 5-stage pipeline to compute the prefix sum of a sequence of values. This technique can further improve the parallelism and throughput of the algorithm [49].
- **DSP+BRAM:** DSP+BRAM is a technique that uses digital signal processing (DSP) blocks and block RAM (BRAM) resources on an FPGA to accelerate the computation of the SHA algorithm. This technique can provide high throughput and low power consumption by leveraging the dedicated hardware resources on the FPGA [50].

4.2 FPGA Implementation Comparoson

Table 2 compares various FPGA implementations of different SHA algorithms, including SHA-1, SHA-256, SHA-384, SHA-512, and SHA-3. The implementations use different techniques such as Carry Select Adder (CSA), Unfold, Parallel Prefix Logic (PPL), 5-Parallel Prefix Logic (5PPL), and

DSP+BRAM. The table provides a comparative analysis of the implementations in terms of their throughput, area utilization, and frequency of operation. This analysis can help designers select the most suitable implementation technique for their specific requirements.

According to Table 2, we can see that the implementation technique used has a significant impact on the throughput and area utilization of the SHA algorithm. For example, the use of 4PPL for SHA-1 on Virtex-6 FPGA achieved a high throughput of 8.607 Gbps but required a higher area utilization of 1230 slices compared to the implementation using Unfold, which had a lower throughput of 1.927 Gbps but required a lower area utilization of 518 slices.

Similarly, for SHA-256, the implementation using PPL on Virtex-5 FPGA achieved a much higher throughput of 12.68 Gbps than the implementation using CSA on the same FPGA, which only achieved a throughput of 1.359 Gbps However, the PPL implementation required a much higher area utilization of 4,793 slices compared to the CSA implementation, which only required 1,203 slices.

For SHA-384, the implementation using CSA on Virtex-5 FPGA achieved a throughput of 0.250 Gbps with an area utilization of 1,914 slices. The implementation using PPL on Virtex-5 FPGA achieved a higher throughput of 1.083 Gbps but required a higher area utilization of 6,606 slices.

For SHA-512, the implementation using CSA on Virtex FPGA achieved a throughput of 0.467 Gbps with an area utilization of 3,792 slices.

Finally, for SHA-3 (Keccak), the implementation using PPL on Virtex-5 FPGA achieved a high throughput of 12.68 Gbps but required a higher area utilization of 4,793 slices. The implementation using PPL on Virtex-6 FPGA achieved a lower throughput of 0.864 Gbps but required a lower area utilization of 393 slices.

Table 2: FPGA implementation of SHA Algorithms

Work		Opt-Tech	FPGA	Area (slices)	Frequency (Mhz)	Throughput (Gbps)
[40]	Sha-1	CSA	Virtex-6	449	213.13	1.347
		Unfold		518	154.35	1.927
		4PPL		1230	172.32	8.607
[41]		PPL	Virtex	950	98.7	2.526
[42]		Unfold	Virtex-2	2894	118	5.9
[43]		4x-Unfold	Virtex2	2394	20.9	0.983
		4x-Unfold+4PPL		4258	41.5	3.541
[44]		Parallelprocess	65nmFPGA	3986	236	14.9
[40]		Pre-Computation	Virtex-6	546	207.90	2.596
[45]	SHA-256	Conrol-Unit	Virtex-5	387	202.54	1.58
[46]	SHA-256	CSA	Virtex	2207	74	0.291
	SHA-384					0.250
	SHA-512					0.467
[47]	SHA-256	CSA	Virtex-5	1,203	170	1.359
[48]	SHA-256	CSA	Virtex-2	1,187	110.10	0.867
		CSA+Pre-Comp		1,274	115.46	0.909
[49]	SHA-512	5PPL	Virtex-2	7,012	54.6	6.989
			Virtex-E	7,151	63.4	9.126
			Virtex-6	7,151	71.3	9.126
			Virtex-7	7,219	91.2	11.674
[50]	SHA-256	DSP+BRAM	Stratix-3	795	205.8	1.621
[51]	SHA-3 (Keccak)	PPL	Virtex-5	4,793	317.11	12.68
[52]		PPL	Virtex-6	393	159	0.864
			Virtex-6	188	285	0.145
[53]		PPL	Virtex-5	1,702	389	18.7
			Virtex-6	1,649	397	19.1
			Virtex-7	1,618	434	20.8
[54]		2PPL	Virtex-4	5,494	269	12.912
			Virtex-5	2,652	352	16.896
			Virtex-6	2,296	391	18.768

4.3 Power Consumption Comparison

In IoT applications, power consumption is a critical factor due to the limited power budget of most IoT devices. Therefore, it is important to consider the power consumption of different cryptographic algorithms when designing FPGA-based IoT systems.This power consumption can vary based on the specific implementation, FPGA device, clock frequency, and optimization techniques used. Conducting power analysis in a real-world scenario is essential to accurately evaluate the power efficiency of different SHA algorithms for IoT applications

In general, SHA-1 is the simplest and fastest algorithm among the SHA family, but it is also the least secure and has been deprecated since 2011. Its power consumption on an FPGA is generally considered to be relatively low compared to the other SHA algorithms.

SHA-2 algorithms are expected to consume more power than SHA-1 due to their longer message digest size and more complex processing steps. The power consumption of SHA-2 on an FPGA will increase with the bit-length of the output hash, the smaller SHA-2 variants (SHA-224 and SHA-256) will consume less power than the larger variants (SHA-384 and SHA-512).

SHA-3 algorithms are expected to consume more power than SHA-1 and SHA-2, this power consumption of SHA-3 will increase with the block size and the bit-length of the output hash. In [17] Kavitha et al compared the power consumption of SHA-2 and SHA-3 on a Xilinx Virtex-7 FPGA. The study found that SHA-3 was about 1.5x more power efficient than SHA-2. Interestingly, the power consumption of SHA-3 was not significantly affected by the number of rounds used in the hash function, while the power consumption of SHA-2 increased significantly with the number of rounds.

Singh et al [18], compared the power consumption of SHA-2 and SHA-3 on a Xilinx Zynq-7000 FPGA. The study found that SHA-3 was about 2x more power efficient than SHA-2. The study also found that the power consumption of SHA-3 was not significantly affected by the message size, while the power consumption of SHA-2 increased with the message size.

4.4 Recommendations for Selecting Hash Functions for IoT Applications

In this section, we provide recommendations for selecting suitable hash functions and FPGA devices for IoT applications based on the performance analysis presented in the previous section. These recommendations are intended to guide IoT developers in selecting the most appropriate hash function and FPGA device for their specific use case.

Firstly, for applications that require high speed, SHA3 may be a suitable option. These hash functions generally achieve high speeds compared to other hash functions.

Secondly, for applications that require low power consumption, SHA-1 or SHA-2 hash functions may be more suitable than SHA-3. SHA-1 and SHA-2 hash functions require less power than SHA-3 due to their simpler algorithms. However, it is important to note that SHA-1 is no longer recommended due to its susceptibility to collision attacks.

Thirdly, for applications that require a balance between speed and resource utilization, SHA-256 or SHA-384 may be suitable options. These hash functions achieve moderate speeds and require moderate resources. As shown in the previous section's table, SHA-384 can achieve good performance with moderate resource utilization.

When choosing an FPGA device, it is important to consider the power consumption, clock frequency, and resource utilization of the device. For example, Xilinx Zynq-7000 and Artix 7 are popular FPGA devices for IoT applications due to their low power consumption and high clock frequency. As shown in the previous section's table, different FPGA devices can have significantly different resource utilization, and the choice of FPGA device can have a significant impact on the performance of the implementation.

It is important to carefully choose the implementation methodology and clock frequency to optimize the performance and power consumption of the FPGA implementation. Increasing the clock frequency can improve the speed of the implementation, but it also increases the power consumption. Therefore, it is important to find the optimal balance between speed and power consumption for the specific use case.

5. SDCURITY ANALYSIS OF SHAALGORITHMS

This section summarizes the known attacks against cryptographic hash functions.

5.1 Collision Attacks

These attacks attempt to find two different input messages that produce the same hash value. If successful, they can allow an attacker to forge a fake digital signature or message that is indistinguishable from a legitimate one.

- **Collision attacks against SHA-1**

Collision attacks against SHA-1 have become increasingly practical and can be executed with relatively low computational effort. In [19] M. Stevens et al presented an improvement to the original SHA-1 collision attack, significantly reducing the computational complexity required to find a collision by a factor of more than 2,000. Their work demonstrated the practical feasibility of finding SHA-1 collisions, raising concerns about the security of systems relying on this hash function.

A novel technique for collision attacks on SHA-1. Published in 2013 by Y. Sasaki et al [20]. The authors proposed an approach called "optimal joint local-collision analysis" and utilized it to discover collisions in SHA-1 with greater efficiency compared to previous attacks. The findings of this study contributed to the mounting evidence of SHA-1's vulnerability.

In [21], M. Stevens et al presented a practical-time collision attack against the AUM cryptographic hash function, which is a variant of SHA-1 used in specific security protocols. The authors demonstrated that the AUM function inherited the vulnerabilities of SHA-1, emphasizing the urgent need for transitioning away from SHA-1 and adopting more secure alternatives.

- **Collision attacks against SHA-2**

Collision attacks against SHA-2 are generally more challenging than those against SHA-1, but there have been

significant advancements in demonstrating practical attacks against SHA-2.

In [22] H. Wu and B. Preneel presented a practical collision attack against SHA-256 and SHA-512. The authors utilized an algorithm called "biclique cryptanalysis" to achieve this feat. The findings highlighted the need for continued scrutiny and potential security enhancements for SHA-2.

Similarly, in [23] Y. Sasak et al extended their collision attack technique, originally applied to SHA-1, to target SHA-2 as well. They employed the "optimal joint local-collision analysis" method to demonstrate collision attacks against SHA-2, indicating potential vulnerabilities in the hash function family.

J. Guo et al [24] introduced a technique for discovering "differential characteristics" in hash functions. The authors specifically applied this method to SHA-1 and SHA-2, enabling the design of more efficient collision attacks. This research contributed to a deeper understanding of the vulnerabilities and potential weaknesses in SHA-2.

Y. Seurin [25] presented a novel collision attack against an 8-round variant of SHA-256. This research significantly reduced the computational complexity required to find a collision by a factor of more than 100, indicating the importance of continued analysis and security evaluations of SHA-2.

- **Collision attacks against SHA-3**

Collision attacks against SHA-3 are generally more difficult than those against SHA-2, primarily due to the sponge construction employed in the SHA-3 hash function. The sponge construction is designed to provide enhanced security against collision attacks.

J. Guo et al [26] presented collision and preimage attacks against round-reduced versions of the SHAvite-3-512 hash function. SHAvite-3-512 was one of the finalists in the NIST SHA-3 competition. The paper demonstrated vulnerabilities in the round-reduced versions of SHAvite-3-512, highlighting the need for careful analysis and evaluation of the security of SHA-3 candidates.

Keccak Team [27] provided a comprehensive analysis of the security of the SHA-3 finalists, including Keccak. The authors conducted an evaluation of the resistance of these functions to various attacks, including collision attacks. The findings contributed to a better understanding of the strengths and weaknesses of the SHA-3 finalists.

Y. Sasaki et al [28] introduced a technique for discovering "path collisions" in the SHA-3 finalists. Path collisions can be utilized to design collision attacks. The authors applied this technique to all of the SHA-3 finalists, further contributing to the analysis and evaluation of their security properties.

It is important to note that these attacks against SHA-3 are generally not practical and do not pose a significant threat to the security of the hash function. SHA-3 is considered a secure choice for most applications requiring a cryptographic hash function, as it underwent extensive scrutiny and evaluation during the NIST SHA-3 competition.

5.2 Length Extension Attacks

These attacks exploit weaknesses in the SHA construction to extend an existing hash value to include additional data, without knowledge of the original input. This can allow an attacker to append additional data to a legitimate message without detection.

- **Length extension attacks against SHA-1**

SHA-1 is vulnerable to length extension attacks, which can be carried out with relatively little computational effort. This vulnerability arises from the fact that SHA-1 uses a Merkle–Damgård construction, which is susceptible to length extension attacks.

In a length extension attack against SHA-1, an attacker who knows the hash of a message can compute the hash of an extended message without knowing the original message. The attacker can do this by taking the hash of the original message and then appending additional blocks to the message along with specially crafted padding. The attacker can then use the final hash value as the initial value for the hash function and continue hashing the appended blocks to produce the hash of the extended message.

In [29], Marc Stevens presented an improved collision attack against SHA-1, a widely used cryptographic hash function. The attack had a complexity of $2^{63.4}$ hash function evaluations, making it the most efficient attack against SHA-1 to date. This attack demonstrated the weakness of SHA-1 and the need to transition to more secure hash functions for cryptographic applications.

Craig Young [30] demonstrated how length extension attacks against SHA-1 can be used to bypass application security measures. Young successfully crafted a malicious transaction that appeared to be legitimate by exploiting a length extension vulnerability in the app's hash function. This attack highlighted the importance of using secure hash functions and implementing proper input validation in applications to prevent such attacks.

The first practical chosen-prefix collision attack against SHA-1, demonstrated by Gaëtan Leurent and Thomas Peyrin [31]. The attack allows an attacker to create two distinct messages with the same hash value. The authors were able to exploit this vulnerability to attack the PGP Web of Trust, a popular key certification system used in email encryption. This attack further emphasizes the importance of transitioning to more secure hash functions for cryptographic applications.

- **Length extension attacks against SHA-2**

SHA-2 is resistant to length extension attacks, which is one of the main reasons why it is considered a more secure hash function than its predecessor, SHA-1. SHA-2 uses a Merkle–Damgård construction, like SHA-1, but includes some additional security features that make it less vulnerable to length extension attacks.

In [32], Hongjun Wu and Bart Preneel presented a theoretical length extension attack against some variants of SHA-2. The attack is based on a variant of the Merkle–Damgård construction called the "secret-prefix" method. The authors show that this attack is feasible against some SHA-2 variants with certain parameters, but they note that it requires a large amount of computation and is not practical in practice. The paper highlights the importance of understanding the

vulnerabilities of hash functions and the need for continued research into developing more secure hash functions.

- **Length extension attacks against SHA-3**

SHA-3 is designed to be resistant to length extension attacks, thanks to its use of the sponge construction. The sponge construction has a built-in mechanism to prevent length extension attacks, which makes it more secure than hash functions based on the Merkle–Damgård construction, such as SHA-1 and SHA-2.

There have been no practical attacks against SHA-3 that exploit its sponge construction to extend the message length. However, SHA-3 is still vulnerable to other types of attacks, such as collision attacks, which have been found against some variants of SHA-3.

5.3 Preimage Attacks

These attacks attempt to find an input message that produces a specific hash value. If successful, they can allow an attacker to generate a message that has the same hash value as a legitimate one, thereby breaking the integrity of the system.

- **Preimage attacks against SHA-1**

Preimage attacks against SHA-1 refer to attacks that allow an attacker to find an input message that produces a givenhash value. In other words, the attacker can find amessage that matches the given hash value, without knowing the original message that produced the hash value. SHA-1 is vulnerable to preimage attacks, particularly for messages with a length less than 2^{63} bits.

In [32], Biryukov and Khovratovich presented a technique for finding preimages of hash functions using algebraic attacks. They applied this technique to SHA-1 and were able to find preimages with a complexity of 2^{51} hash operations. This attack demonstrated the vulnerability of SHA-1 to preimage attacks and the need for stronger hash functions.

Stevens et al. [33] presented an improved version of the Wang attack that reduces the complexity of finding a preimage of SHA-1 to 2^{52} hash operations. They also presented a new technique for finding a preimage of SHA-1 based on a technique called message modification. This research highlights the need for stronger hash functions and the importance of continued research into improving the security of cryptographic algorithms.

- **Preimage attacks against SHA-2**

Preimage attacks against SHA-2 are attacks where an attacker tries to find a message that hashes to a specific hash value produced by SHA-2. Although SHA-2 is considered to be more secure than SHA-1, there have been several attacks on SHA-2 that can be used to perform preimage attacks. Mendel et al. [34] presented an improved boomerang attack on SHA-2 that reduced the complexity of finding a preimage for SHA-256 to 2^{251} hash operations. This research demonstrated the vulnerability of SHA-2 and the need for continued research into developing more secure hash functions.In [35], Peyrin and Seurin presented a linearization attack on SHA-2 that allowed the authors to find a preimage for SHA-256 with a complexity of 2^{236} hash operations. This attack demonstrated the need for more secure hash functions and the importance of continued research into developing stronger cryptographic algorithms.The rebound attack on SHA-2, introduced by Mendel et al. [36] , which can be used to find preimages for SHA-256 with a complexity of 2^{254} hash operations. This attack highlighted the need for stronger hash functions and the importance of continued research into developing more secure cryptographic algorithms.In 2017, Morawiecki and Srebrny [37] presented a meet-in-the-middle attack on SHA-2 that allowed the authors to find a preimage for SHA-256 with a complexity of 2^{255} hash operations. This attack demonstrated the vulnerability of SHA-2 and the need for continued research into developing more secure hash functions.It is worth noting that while these attacks demonstrate vulnerabilities in SHA-2, the complexity of the attacks is still extremely high and currently considered to be infeasible for practical purposes.

- **Preimage attacks against SHA-3**

SHA-3 was designed to resist preimage attacks by using a sponge construction and a permutation function based on the Keccak algorithm, which has strong security properties.

Wang et al. [38] analyzed the security of the SHA-3 finalists, including Keccak, and found no practical preimage attacks. This research demonstrated the strength and security of the SHA-3 finalists and their resistance to preimage attacks.

In [39], Biryukov and Khovratovich presented a linear attack on round-reduced Keccak, but did not result in a practical preimage attack. This research highlighted the need for continued analysis and research into the security of Keccak and other cryptographic algorithms.

6. CONCLUSION

The selection of a secure hash algorithm (SHA) for IoT devices requires a careful consideration of the trade-offs between security and performance. The paper presents a detailed comparison of SHA-1, SHA-2, and SHA-3 based on their QoS parameters, including resource utilization, processing speed, power consumption, and resistance against different attacks. The results show that SHA-1, while widely used in many legacy systems, is no longer recommended due to its susceptibility to collision attacks. SHA-1's fixed 160-bit output size and 512-bit message block size limit its flexibility and make it vulnerable to attacks. In contrast, SHA-2 provides a good balance between security and performance, making it a suitable choice for many IoT applications. SHA-2's variable output size and message block size, ranging from 224 to 512 bits, and a fixed message block size of 512 bits, offer greater flexibility and make it more resistant to attacks such as collision and length extension. SHA-3, the newest SHA algorithm, offers some advantages over SHA-2, such as its resistance to length extension attacks and its flexibility in output size. SHA-3 has a variable message block size that ranges from 1152 to 576 bits depending on the hash output length, which can make it more efficient in some cases. However, SHA-3's lower number of rounds and larger message block size can make implementation more complex and resource-intensive in some cases.

REFERENCES

1. N. Meghanathan. **IoT security threats and challenges: Current scenario, security threats and privacy issues in the IoT and its solutions**, in International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2017, pp. 2255-2260.
2. Xiong, S. Cheng and K. Yang. **Secure Message Transmission in Internet of Things Based on Message Digest**, in International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, 2018, pp. 122-126,.
3. R. Singh, A. Passi and G. Kaur. **Security Issues and Solutions in Internet of Things: A Comprehensive Study**, in 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, 2018, pp. 229-233.
4. S. Gao, L. Xu and J. Ma. **Security of Firmware Updates in Internet of Things: Challenges and Solutions**, in International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), Beijing, 2016, pp. 104-109.
5. P. Phu, T. T. Duc and P. Cong. **Secure Boot in IoT Devices: Challenges and Solutions**, in International Conference on Advanced Technologies for Communications (ATC), Quy Nhon, 2017, pp. 62-67.
6. A. Alghamdi A., & Hussain, M**. A Survey on Secure Hash Algorithm (SHA) and its Variants**, Journal of Network and Computer Applications, vol. 128, pp. 1-26, 2019.
7. Afzal, M. K., & Ahmad, I. **A Survey of Security Challenges and Solutions in IoT-based Healthcare Systems**, Journal of Ambient Intelligence and Humanized Computing, vol. 11(8), pp. 3073-3096, 2020.
8. Singh, A., & Sharma, S. K. **Secure IoT Communication Using SHA-512**, In Proceedings of the International Conference on Computational Intelligence and Data Science, 2018, pp. 190-196.
9. Gao, J., & Li, L. **Research on IoT Security Based on SHA-3 Algorithm**, Journal of Physics: Conference Series, 1176, 042020, 2019.
10. Zhang, Y., Zhang, H., & Li, Z. **A Secure and Efficient IoT Communication Protocol Based on SHA-256 and ECC**, IEEE Access, vol. 8, pp. 115079-115088, 2020.
11. Bao, W., Li, J., & Liu, Y. **A Lightweight Secure Communication Protocol for IoT Based on SHA-256 and ChaCha20**. IEEE Internet of Things Journal, vol. 6(1), pp. 55-65, 2019.
12. Wang, X., Li, C., & Zhou, Y. **A Secure and Efficient IoT Communication Protocol Based on SHA-256 and AES**, Journal of Ambient Intelligence and Humanized Computing, vol. 12(10), pp. 10259-10271, 2021.
13. Liu, Y., Li, J., & Bao, W. **A Secure and Efficient IoT Communication Protocol Based on SHA-256 and SM4**. IEEE Internet of Things Journal, vol. 7(6), pp. 5525-5534, 2020.
14. Vasilakos, A. V., & Sun, Q. **Security Challenges and Opportunities in Industrial Internet of Things**, IEEE Transactions on Industrial Informatics, vol. 13(2), pp. 686-689, 2019.
15. Wang, X., Yan, B., & Sun, G. **A Secure and Efficient IoT Communication Protocol Based on SHA-256 and SM2**. Journal of Ambient Intelligence and Humanized Computing, vol. 10(7), pp. 2547-2558, 2019.
16. J. Raja and G. K. Sivakumar**. IoT Security using SHA3 Algorithm**, in International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2018, pp. 1-5.
17. N. Kavitha, A. Karthikeyan, and P. Dananjayan. **Comparison of the Power Consumption of SHA-2 and SHA-3 on FPGAs**, in International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2017, pp. 0553-0557.
18. J. Singh and A. K. Singh, **Power Consumption Comparison of SHA-2 and SHA-3 on FPGAs**, in 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 1-6,.
19. Marc Stevens, Arjen Lenstra, and Benne de Weger. **Freestart Collision on Full SHA-1**. in Advances in Cryptology – EUROCRYPT 2013. Springer, 2013.
20. Yu Sasaki, Lei Wang, and Kazumaro Aoki. **New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis**, in Advances in Cryptology – EUROCRYPT 2013. Springer, 2013.
21. Marc Stevens, Pierre Karpman, and Thomas Peyrin. **A Practical-Time Attack on the AUM Cryptographic Hash Function**, in Advances in Cryptology – ASIACRYPT 2016. Springer, 2016.
22. Hongjun Wu and Bart Preneel. **Cryptanalysis of SHA-256 and SHA-512**, in EUROCRYPT 2009. Springer, 2009.
23. Yu Sasaki, Lei Wang, and Kazumaro Aoki. **New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis**, in EUROCRYPT 2013. Springer, 2013.
24. Jian Guo, San Ling, and Christian Rechberger. **Finding SHA-1 Characteristics: General Results and Applications**, in EUROCRYPT 2013. Springer.

25. Yannick Seurin. **A New Meet-in-the-Middle Collision Attack against 8-Round SHA-256**, in CRYPTO 2016. Springer.
26. Jian Guo, Thomas Peyrin, and Axel Poschmann. **Collision and Preimage Attacks on the Round-Reduced SHAvite-3-512 Hash Function**, in Selected Areas in Cryptography - SAC 2012. Springer.
27. Keccak Team. **Cryptanalysis of SHA-3 Candidates**, in Cryptology ePrint Archive, Report 2011/102. 2011.
28. Yu Sasaki, Lei Wang, and Kazumaro Aoki. **Finding Path Collision in the SHA-3 Candidates**, in Advances in Cryptology - ASIACRYPT 2011. Springer.
29. Marc Stevens. **Improved Collision Attack on SHA-1**, in Advances in Cryptology - EUROCRYPT 2013. Springer.
30. Craig Young. **The SHAppening: Cryptographic Hashes and Their Impact on App Security**, in Black Hat USA, 2015.
31. Gaëtan Leurent and Thomas Peyrin. **SHA-1 is a Shambles: First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust**, in Advances in Cryptology - EUROCRYPT 2020. Springer, 2020.
32. Biryukov, A., & Khovratovich, D. **Improving the preimage attack on SHA-1**. in Advances in Cryptology – ASIACRYPT 2009, Springer Berlin Heidelberg, 2009, pp. 57-75.
33. Stevens, M., Bursztein, E., Karpman, P., & Albertini, A. **The first collision for full SHA-1**. in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2017, pp. 372-387.
34. Mendel, F., Pramstaller, N., & Rechberger, C. **Improved boomerang attacks on reduced SHA-256**, International Conference on Cryptology and Network Security, Springer, Cham ,pp. 101-119..
35. Peyrin, T., & Seurin, Y. **A linearization attack on SHA-256**, in Advances in Cryptology – EUROCRYPT 2011, Springer Berlin Heidelberg, 2011, pp. 340-357.
36. Mendel, F., Pramstaller, N., & Rechberger, C. **The rebound attack on reduced SHA-256**, in Annual Cryptology Conference, Springer, Berlin, Heidelberg, 2015, pp. 260-287.
37. Morawiecki, P., & Srebrny, M. **Meet-in-the-middle preimage attacks on SHA-2**, in International Conference on Cryptology and Network Security, Springer, Cham, 2017, pp. 84-101.
38. .Wang, L., Wu, H., Hasan, A., & Saso, D. **Preimage attack on hash functions and security of SHA-3 candidates**, Information Security Journal: A Global Perspective, vol. 21(1-2), pp. 38–52, 2012.
39. Dmitry Khovratovich and Alex Biryukov. **Linear Cryptanalysis of Round-Reduced Keccak**. in Fast Software Encryption. Springer, Berlin, Heidelberg, 2011.
40. R. K. Makkad and A. K. Sahu. **Novel design of fast and compact SHA-1 algorithm for security applications**, in Proceedings of the IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT'16), 2016, pp. 921–925.
41. A. P. Kakarountas, G. Theodoridis, T. Laopoulos, and C. E. Goutis. **High-speed FPGA implementation of the SHA-1 hash function**, in Proceedings of the IEEE Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2005, pp. 211–215.
42. Eun-Hee Lee, Je-Hoon Lee, Il-Hwan Park, and Kyoung-Rok Cho. **Implementation of high-speed SHA-1 architecture**. IEICE Electron. Expr. 6, 16, pp. 1174–1179, 2009.
43. Yong Ki Lee, Herwin Chan, and Ingrid Verbauwhede. **Throughput optimized SHA-1 architecture using unfold ing transformation**. in Proceedings of the International Conference on Application-specific Systems, Architectures and Processors (ASAP'06), 2006. IEEE, pp. 354–359.
44. T. Isobe, S. Tsutsumi, K. Seto, K. Aoshima, and K. Kariya. 2010. **10 Gbps implementation of TLS/SSL accelerator on FPGA**. in Proceedings of the IEEE 18th International Workshop on Quality of Service (IWQoS'10) , 2010. pp. 1–6.
45. Hassen Mestiri, Fatma Kahri, Belgacem Bouallegue, and Mohsen Machhout. 2014. **Efficient FPGA hardware implementation of secure hash function SHA-2**. Int. J. Comput. Netw. Inf. Sec. Vol.7, (1) 2014.
46. Wanzhong Sun, Hongpeng Guo, Huilei He, and Zibin Dai. **Design and optimized implementation of the SHA- 2 (256, 384, 512) hash algorithms,** in Proceedings of the 7th International Conference on ASIC (ASICON'07). IEEE, 2007, pp. 858–861.
47. Anane Mohamed and Anane Nadjia. **SHA-2 hardware core for Virtex-5 FPGA**. In Proceedings of the 12th Inter- national Multi-Conference on Systems, Signals & Devices (SSD'15), IEEE, 2015, pp. 1–5.
48. Ignacio Algredo-Badillo, C. Feregrino-Uribe, René Cumplido, and Miguel Morales-Sandoval. **FPGA-based im- plementation alternatives for the inner loop of the secure hash algorithm SHA-256**, Microprocessor andMicrosystem journal . Vol. 37(6), pp. 750–757,2013.
49. George S. Athanasiou, Harris E. Michail, George Theodoridis, and Costas E. Goutis **Optimising the SHA-512 cryptographic hash function on FPGAs**, IET Comput. Dig. Techniques, vol. 8(2), pp. 70–82. 2013.
50. Rabia Shahid, Malik Umar Sharif, Marcin Rogawski, and Kris Gaj. **Use of embedded FPGA resources in im-**

plementations of 14 round 2 sha-3 candidates, in Proceedings of the International Conference on Field-Programmable Technology (FPT'11). IEEE, 2011, pp. 1–9.

51. Hassen Mestiri, Fatma Kahri, Mouna Bedoui, Belgacem Bouallegue, and Mohsen Machhout**. High throughput pipelined hardware implementation of the KECCAK hash function**, In Proceedings of the International Symposium on Signal, Image, Video and Communications (ISIVC'16), IEEE, 2016, pp. 282–286.
52. Bernhard Jungk and Jurgen Apfelbeck. **Area-efficient FPGA implementations of the SHA-3 finalists**, in Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'11). IEEE, 2011, pp. 235–241.
53. George S. Athanasiou, George-Paris Makkas, and Georgios Theodoridis. **High throughput pipelined FPGA im- plementation of the new SHA-3 cryptographic hash algorithm**, in Proceedings of the 6th International Symposium on Communications, Control and Signal Processing (ISCCSP'14). IEEE, 2014, pp. 538–541.
54. Lenos Ioannou, Harris E. Michail, and Artemios G. Voyiatzis. **High performance pipelined FPGA implementation of the SHA-3 hash algorithm**, In Proceedings of the 4th Mediterranean Conference on Embedded Computing (MECO'15). IEEE, 2015, pp. 68–71.