EVOLUTION AND ANALYSIS OF SECURE HASH ALGORITHM (SHA) FAMILY

Burhan Ul Islam Khan[1*], Rashidah Funke Olanrewaju[2], Malik Arman Morshidi[3], Roohie Naaz Mir[4], Miss Laiha Binti Mat Kiah[5] and Abdul Mobeen Khan[6]

[1,2,3]Department of ECE, KOE, International Islamic University Malaysia (IIUM), Kuala Lumpur, 50728, Malaysia

[4]Department of Computer Science and Engineering, National Institute of Technology (NIT), Srinagar, 190006, India

[5]Department of Computer System & Technology, Universiti Malaya (UM), Kuala Lumpur, 50603, Malaysia

[6] ALEM Solutions and Technologies, Aspen Commercial Tower, Dubai, 11562, United Arab Emirates

Email: burhan.iium@gmail.com[1*] (corresponding author), frashidah@iium.edu.my[2], mmalik@iium.edu.my[3], naaz310@nitsri.net[4], misslaiha@um.edu.my[5], consultmobeen@gmail.com[6]

DOI: https://doi.org/10.22452/ mjcs.vol35no3.1

ABSTRACT

With the rapid advancement of technologies and proliferation of intelligent devices, connecting to the internet challenges have grown manifold, such as ensuring communication security and keeping user credentials secret. Data integrity and user privacy have become crucial concerns in any ecosystem of advanced and interconnected communications. Cryptographic hash functions have been extensively employed to ensure data integrity in insecure environments. Hash functions are also combined with digital signatures to offer identity verification mechanisms and non-repudiation services. The federal organization National Institute of Standards and Technology (NIST) established the SHA to provide security and optimal performance over some time. The most well-known hashing standards are SHA-1, SHA-2, and SHA-3. This paper discusses the background of hashing, followed by elaborating on the evolution of the SHA family. The main goal is to present a comparative analysis of these hashing standards and focus on their security strength, performance and limitations against common attacks. The complete assessment was carried out using statistical analysis, performance analysis and extensive fault analysis over a defined test environment. The study outcome showcases the issues of SHA-1 besides exploring the security benefits of all the dominant variants of SHA-2 and SHA-3. The study also concludes that SHA-3 is the best option to mitigate novice intruders while allowing better performance cost-effectively.

Keywords: Secured Hash Algorithms, Message Digest, Cryptographic Hashing, Statistical Analysis, Fault Analysis

1.0 INTRODUCTION

The Internet is more critical in the current era than ever before, and almost everyone uses Internet services for different purposes. In most cases, the data traversing over the Internet comprises private or concealed information that everybody wants to protect [1], [2]. The rapid advancement in technologies has raised significant concerns about protecting our data from unauthorized access. With advanced software programs, malicious users can intercept data in transit or breach the confidentiality of data stored on distributed storage systems such as the cloud [3]. Therefore, protecting its information is pivotal in safeguarding any organization or individual's company and client assets. Security usually means using the best preventive and defensive measures to protect data from unauthorized access [4]. Cryptography is the science of security applications that offers a mechanism for keeping users' information secure with privacy and confidentiality [5]. As an imperative sub-area of fast communication, cryptography allows a protected communication process between different users so that malicious users cannot access the contents inside the file [6]. However, this secure communication area has a long history of achievements and failures. Several encoding messages have appeared over the eras and are continually broken after a while [7]. A cryptographic hash algorithm aims to provide secure communication using the digest of messages to generate a hash value of data to detect unauthorized attempts to data. Hash algorithms are widely used in many applications, such as data integrity and corruption verification, ownership protection, authentication, and many more [8]. The system of cryptocurrency also depends on the hashing algorithms and functions.

The hashing algorithms are vital cryptographic primitives that accept input, process it and generate a digest [9]. The digest is a fixed-size alphanumeric string that humans cannot understand. Some older and newer versions of cryptographic hash functions are available such as the Message-Digest Algorithm 5 (MD5), MD4, SHA family, and many other hashing functions [10]. Hash is a computationally secure form of compression concerning security aspects [11]. There are different variants of SHA witnessed to date, e.g., SHA-1, SHA-2, SHA-256, SHA-348, and SHA-512. With the availability of discussion carried out by various research works and legal authorities, e.g., NIST, there are constantly evolving versions of secured encryption techniques [12], [13]. The research in this direction resulted in the formation of SHA-3, a new version. However, owing to the novel nature of the structure of this encryption technique, the degree of strength and weakness of this algorithm is still unknown.

At present, there are various methods where SHA variants have been reportedly used for catering to different security demands, e.g., authentication of the electronic document [14], improving security system using chaotic map [15], secured message hiding [16], image security [17], improving anonymity [18], improving security by combining hashing with genetic science [19], strengthening security via blockchain [4], [20], authentication over the mobile network [21]. Apart from this, there is also literature [22], [23] on evaluating the strength of SHA. However, there is no explicit discussion to assess the strength of the three most essential SHA family members.

The prime objective of this paper is to present a comparative assessment of SHA algorithms besides emphasizing the adoption of SHA-3. The specific goals of the paper are:

- Assess the effectiveness of all the SHA versions over a similar test environment to chalk out a certain level of inference, and
- Perform statistical analysis, performance analysis and extensive fault analysis to assess the strength of SHA approaches.

The paper adopts a simple experimental design methodology considering three variants as well as sub-classes of SHA, i.e. SHA-1, SHA-2, and SHA-3. SHA-3 is exhibited to perform well in contrast to its counter-SHA1 and SHA2 versions with respect to minimal randomness of hashes, increased clock per byte, and maximal recovery of bits. Unlike any existing investigations, the proposed study contributes towards a generalized mathematical approach considering the standardized evaluation method to prove that SHA-3 is a robust and well-performing algorithm in a contemporary state. This outcome increases the higher adoption rate for any form of application that demands optimal security measures.

The remaining sections of the paper are organized as follows: Section 2 provides a brief discussion on the hashing operation, while it is more elaborately discussed concerning its all-around aspect in section 3 with respect to the significant SHA variant. The proposed manuscript evaluates all the multiple variants of SHA via statistical analysis discussed in section 4, performance analysis addressed in section 5, and extensive fault analysis discussed in section 6. The outline of the study outcome is briefed in section 7, while section 8 provides conclusive remarks. References are listed at the end.

2.0 STUDY BACKGROUND

The adoption of hashing mechanism is an integral part of the majority of security application designs. Technically, a hash function can be defined as a mathematical function that converts a numerical input value into another compressed numerical value. An arbitrary length is considered the input of a hash function, while a fixed length always characterizes hashing. Fig. 1 highlights the usage of messages of random length, which results in a fixed-length hash value after being subjected to a hash function.

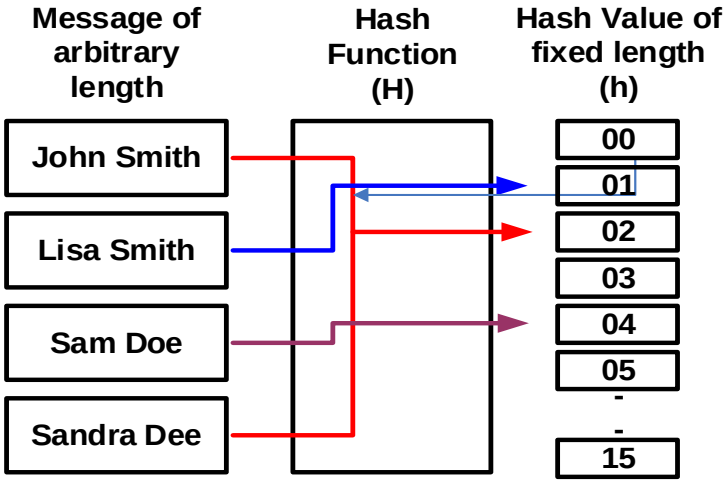


Fig. 1: Conventional Hashing Mechanism

To develop it as an efficient cryptographic tool, specific essential properties must feature a standard hash function. The first important property is preimage resistance, which induces a computational challenge to reverse the hash function operation [24]. This property is essential as it safeguards from an intruder who has possession of hash value only and attempts to obtain the input data. The second important property is called second preimage resistance, which ensures a higher degree of complexity in getting hash from different inputs for a given input and hash [25]. This security property of hash ensures safety from an intruder who has both input value and its respective hash value.

In contrast, the intruder is interested in replacing the alternative value as a legitimate value in the position of the source input value. The third property of hashing is Collision Resistance, which ensures complexity in obtaining two different input values of any length yielding the same hash [26]. This property provides that the collision is highly complex, making it difficult for the intruder to explore two input values to possess the same hash. Using these security properties makes it feasible to construct multiple security applications [27], [28]. One of the essential applications of hashing is to carry out password storage, while another frequently adopted application is data integrity. Most hashing can be utilized for authentication systems because the aforementioned two standards are frequently employed in applications [29].

The construction mechanism of hashing is also quite simplified as well as unique. Referring to Fig. 1, it can be seen that a mathematical function is quite essential to generating hashes. The sample message of arbitrary length, i.e., "John Smith," "Lisa Smith," "Sam Doe" and "Sandra Dee," will formulate blocks of data that further result in hash codes as an outcome. Usually, the size of such a data block resides in a range of 128-512 bits. At the same time, the hash function of different types constitutes to become a hashing algorithm. The processing of hashing operation consists of multiple rounds similar to block cipher. The process rounds continue until the complete messages are subjected to hash.

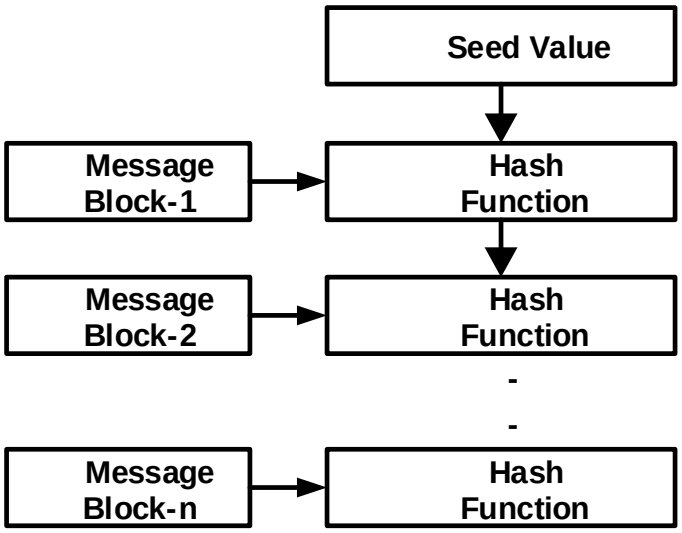


Fig. 2: Construction Process of Hash

The process flow shown in Fig. 2 is also called as avalanche effect of hashing [30]. Due to the fact that the hash value associated with the first block of the message acts as an input file for the second hash operation and impacts the consecutive function, this effect generates unique hash values for messages without any form of mapping relation with each other. As a result, the two hash values differ by at least one bit of data. The construction process of hashing also has unique differences between hash algorithms and hash functions. The hash algorithm is defined as a complete process that formulates breaking down the message block. It also establishes the result obtained from the previous block of the message and their connectivity with the other generated blocks of messages.

The hash function generates a hash code subjected to multiple blocks associated with a fixed-length of binary data as an outcome. Therefore, a hash function needs to map the anticipated input over its range of outcomes as evenly as feasible. The robustness of hashing is carried out considering theoretical and practical approaches. Theoretical methods will mainly compute the probability of mapping all the keys in a single slot, whereas the practical process will consist of assessing the longest probe sequence. Usually, the practical method consists of uniform hashing, which means that any key value can be used to map specific slots with maximum probability. However, such probability of a real-time attack is significantly small, and hence, hashing is one of the cost-effective mechanisms adopted in network security.

3.0 SHA FAMILY

The adoption of hashing is witnessed in improving security over different application variants [31], [32], [33], [34], [35]. The hashing algorithm effectively identifies and validates that a legal person forwards the data received by a user, or the received information is authentic and not altered [36]. Many algorithms have been introduced to generate hash values, some of which were rejected, and some have become standards. Message Digest (MD) and SHA (SHA-1 and SHA-2) are the popularly accepted standards for generating hash values [37], [38]. It is always necessary to upgrade to a new technique or replace an existing one to meet the latest requirements with technology developments. A hash algorithm is an explicit cryptographic operation that alters an arbitrary-length input message to a fixed compressed numerical value called a hash value. The fundamental attributes of the hash algorithms are that it is difficult to find two different messages with the same hash value [39]. A typical process of the hash algorithm is exhibited in Fig. 3.

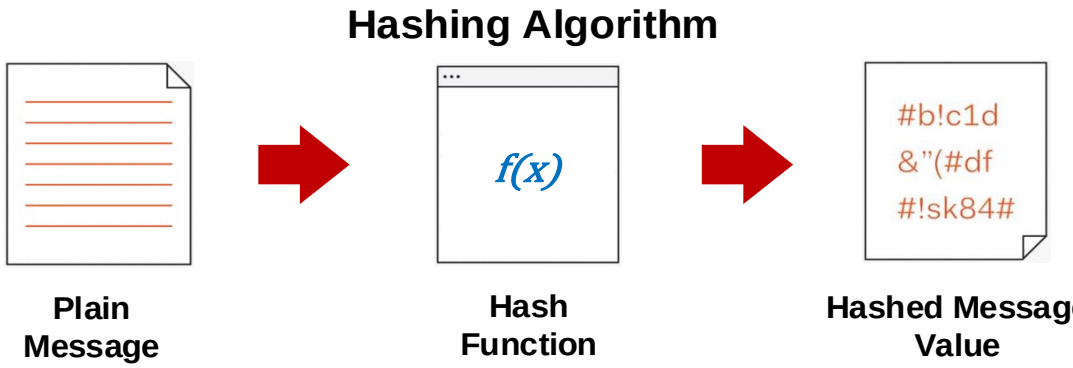


Fig. 3: A Typical Process of Hash Algorithm

A hash function is a mathematical operation that takes in plain message data of variable length and provides a fixed size of the hash message value. Numerically, the process of a hash algorithm can be expressed as follows:

$$H(fx): \{0,1\}^* \rightarrow \{0,1\}n \quad (1)$$

In expression (1), $\{0,1\}^*$ indicates the set of arbitrary-length elements, including the empty string, and $\{0,1\}n$ refers to elements with length n. Therefore, a hash function maps a set of fixed-length elements to arbitrary-length elements. However, the length of the output hashed values mainly depends on the type of hash algorithm used. Generally, the size of the hash algorithms ranges between 160 bits and 1088 bits. To obtain a preset fixed-length hash value, the input message must be divided into fixed-size data blocks because the hash function receives data with a fixed length, as shown in Fig. 4.

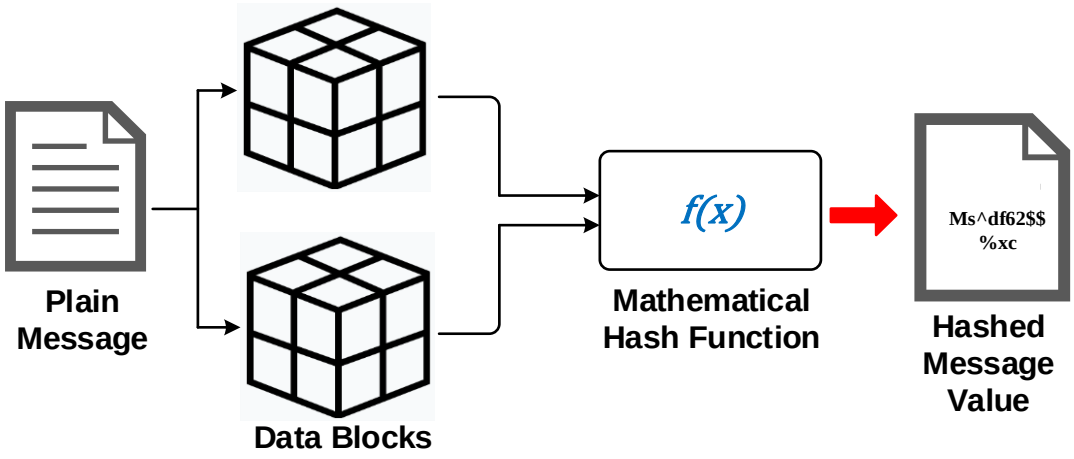


Fig. 4: A Typical Process of Fixed-Size Data Blocks

The data block sizes vary depending on the type of algorithm used. For instance, let's consider the case of SHA-1, which takes plain text messages in the block size of 512 bits. If the plain text message's length is 512-bit, the hash function $H(fx)$ executes 80 rounds (80 rounds means one-time execution). If the plain text message's length is 1024-bit, then the message will be divided into two fixed-length block sizes of 512-bit and the hash function executing two times. However, the size of plain text messages is rarely in multiple of 512-bit. A technique called padding is considered to divide input plain text messages into fixed-length data blocks for other cases. The process of the fixed-length block is demonstrated in Fig. 4. The plain text message is first divided into multiple fixed-length block sizes, and the output of the first data block and the consecutive data block is fed to the hash function. Therefore, the final output is the shared value of all blocks. If the alteration of one bit anywhere in the text message, the overall hashed value will be changed, called the avalanche effect [40], [41].

3.1 Properties of Hash Function

A hash function should satisfy the following properties [42], [43].

- Resistance to Collision Attacks: This attribute indicates that two different messages should not have the same hash value. An adversary can't identify the same hash value $H(m)$ for two messages (m_1, m_2). Collision attacks refer to a process when a pair of different messages with the same hash $H(m)$ conflict attacks occur. The hash function must have the exact property of not constructing the same hashed value for two different messages.
- Resistance to Preimage Attacks: This property refers to the function of a hash algorithm strong enough that it will be challenging to generate the corresponding message for a given hash value. Therefore, the hash function should have preimage resistance. Preimage refers to a message that hashed to a provided value. This attack generally considers that at least one message is hashed to a given value. Therefore, an adversary cannot obtain the original data from the given hash value.
- Resistance to Second Preimage Attacks: Given a message, it must not be possible to identify another message that will construct the same hash value as the previous message. Therefore, the hash function must have the ability to resist the second preimage attack. A second preimage refers to a message that hashes to the same value as a given message.

3.2 Secure Hash Algorithms

SHA is a cluster of hash mathematical functions released by NIST as a US Federal Information Processing Standard. NIST first proposed the SHA-1 hash algorithm in 1995. The design principle of this algorithm is based on Merkle Damgard's structure as MD5 (Fig. 5). The algorithm uses a string of any length and provides a 160-bit compressed MD for variable-length input messages. Usually, in this scheme, the message is padded with 1; the required 0's are added to make the message length equal to 64 bits (less than an even multiple of 512). Sixty-four bits representing the length of the original message are added to the end of the padded message, which is processed in 512-bit data blocks. The following are the functional steps involved in SHA-1: The first step is padding bits to the end of the message length. The next step is subjected to the process of length appending. The third step is about dividing the input message into 512-bit data blocks. In this step of the hashing algorithm, chaining variables are initialized, i.e., internal state size. This means that number of bits is carried over to the next block. Fig. 5 shows the operation of SHA-1.

In SHA-1, five chaining variables of 32 bits are taken, i.e., each equals 160 bits of the total. The final step is block processing. In this step, chaining variables are copied, and the data block of 512 bits is divided into 16 sub-blocks processed for 80 rounds for single-time execution. SHA-1 is used in a wide range of security applications and protocols, including Transport Layer Security and SSL, PGP, SSH, S/MIME and IPsec. However, this hash algorithm is vulnerable to collision attacks due to cryptographic flaws. In the past few years, confirmed cases of collision attacks have been reported for this algorithm. As a result, after 2010, most encryption users no longer recognize this standard.

NIST introduced the SHA-2 hashing algorithm in 2002 with hash code lengths of 224, 256, 348 and 512. The design principle of this algorithm is based on the similar approach of the MD5 and SHA-1. However, this algorithm is more robust than SHA-1 due to the inclusion of a nonlinear function in the compression module [31]. Fig. 6 shows the operation of SHA-2.

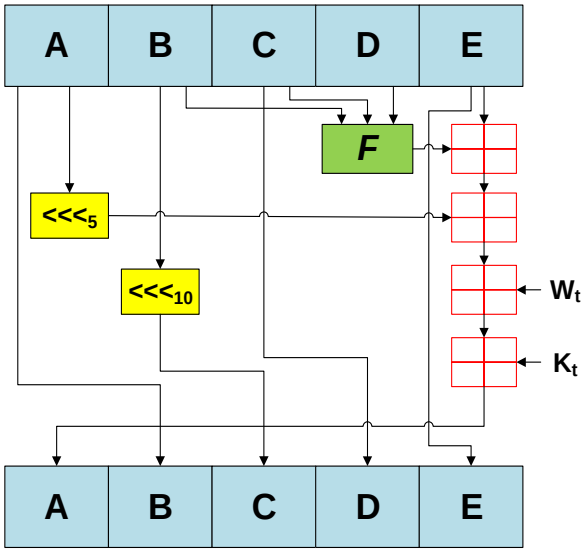


Fig. 5: One SHA-1 Round [37]

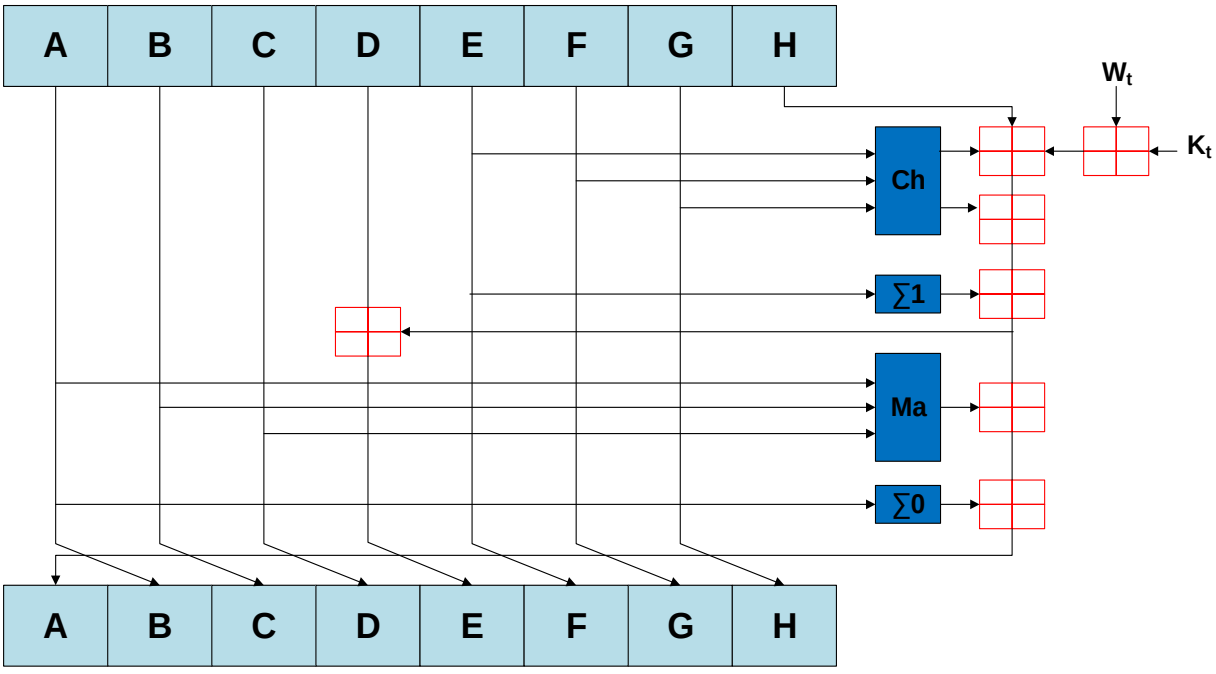


Fig. 6: One SHA-2 Round [22]

In SHA-2, the message is padded with 1; the required number of 0's are added to make its length equal to sixty-four bits (less than an even multiple of 512). Sixty-four bits representing the length of the original message are added to the end of the padded message, which is processed in 512-bit data blocks. Each data block contains 1024 bits as a sequence of 64-bit words and supports SHA-512 and SHA-384. However, SHA-2 is not preferred to ensure integrity, as it has less time-efficiency than SHA-1 due to the absence of multithreading. Besides, SHA-256 for hash cash is primarily used in cryptocurrency to achieve security on the transactions among peers in the Bitcoin network. Both SHA-512 and SHA-256 are new functions that use different constants and shift amounts and differ in the number of rounds. A research study carried out has reported that SHA-2 is vulnerable to attacks. Following is the overview of the computing steps involved in SHA-2:

- The padded length - The total message length to be hashed should be a multiple of 1024 bits.

- The padding is initiated by adding 1 as the first bit followed by the required number of 0's with 128 bits message $\{m_1, m_2 \dots m_n\}$
- Initial hash value, $H^{(0)}\ to\ H^{(i)} = H^{(i-1)} + C_M^{(i)}\left(H^{(i-1)}\right)$
- Where C indicates compression function and $H^{(N)}$ is the hash value of the message (m).
- The output obtained from SHA-256 is the 64 bits of a message, and the six logical attributes function as a, b, c with a 64-bit message.

After a successful collision attack in SHA-1, an open competition was announced by NIST to develop SHA-3 (a new hash algorithm). A few years later, on October 2, 2012, the Keccak (as SHA-3) was announced as the competition winner. In 2015, NIST recognized SHA-3 as a standard hash function [32], [33]. Its operation is shown in Fig. 7.

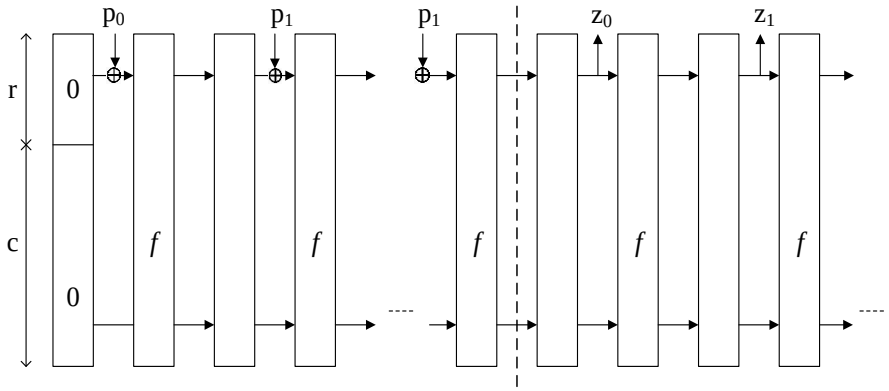


Fig. 7: One SHA-3 Round [36]

SHA-3 utilizes a sponge structure that consists of two phases, the first is the absorbing phase, and the second is the squeeze phase. The message block should be XORed into sub-space in the absorption phase and converted. In the squeezing step, output blocks are analyzed from the same sub-space and transformed with state transitions. The different variants of SHA-3 consisting of SHA-3:224, SHA-3:256, SHA-3:348, SHA-3:512 generate 224, 256, 348, and 512-bit message digests, respectively. It uses a block size of 1152, 1088, 832 and 576 bits, respectively.

This algorithm shows relatively low software performance compared with other hash functions. The comparative analysis of different SHA variants is demonstrated in Table 1 and Table 2 with their essential parameters. Table 1 presents a comparative analysis of different secure hashing algorithms. The study is carried out concerning the functional parameters of each technique discussed. In Table 2, a comparative analysis is given concerning the security aspect and computational parameters.

Table 1: Comparative Analysis of SHA concerning their parameters

Secure Hashing Algorithm		Output Size (Bits)	Internal State Size (Bits)	Block Size (Bits)	Rounds	Operation
SHA-1		160	160(5x32)	512	80	AND, XOR, OR, Add(mod 232), Rotate with no carry
SHA-2	SHA-224	224	256(8x32)	512	64	AND, XOR, OR, Add(mod 232), Rotate with no carry, Right Logical Shift
	SHA-256	256				
	SHA-384	384	512(6x64)	1024	80	AND, XOR, OR, Add(mod 264), Rotate with no carry
	SHA-512	512				
	SHA-512/224	224				
	SHA-512/256	256				

SHA-3	SHA-3-224	224	1600(5x5x64)	1152	24	AND, XOR, NOT, Rotate with no carry
	SHA-3-256	256		1088		
	SHA-3-512	384		832		
	SHA-3-512	512		576		
	SHAKE128	d(arbitrary)		1344		
	SHAKE256	d(arbitrary)		1088		

Table 2: Comparative Analysis of SHA on Security and Computational Performance

Secure Hashing Algorithm		Security (in Bits Against Collision Attack)	Capacity Against Extensive Length Attacks	Performance on Skylake		Year of Release
				Long Message	8 bytes	
SHA-1		<63 (collusion found)	0	3.47	52	1995
SHA-2	SHA-224	112	32	7.62	84.50	2004
	SHA-256	128	0	7.63	85.25	2001
	SHA-384	192	128(≤384)	5.12	135.75	2001
	SHA-512	256	0	5.06	135.50	
	SHA-512/224	112	288	5.12	135.75	2012
	SHA-512/256	128	256	5.12	135.75	
SHA-3	SHA-3-224	112	448	8.12	154.25	2015
	SHA-3-256	128	512	8.59	155.50	
	SHA-3-384	192	768	11.06	164.00	
	SHA-3-512	256	1024	15.88	164.00	
	SHAKE128	min(d/2,128)	256	7.08	155.25	
	SHAKE256	min(d/2,256)	512	8.59	155.50	

3.3 Issues in SHA-1 and SHA-2

SHA-1 and SHA-2 were both introduced by the NSA and issued as patents for public use. Although SHA-1 and SHA-2 are not identical, they are designed based on the same mathematical concept, which comprises the same flaws. However, the reason that makes SHA-2 more secure than SHA-1 is that SHA-2 uses more considerable input and output, and it generates hash values with increased length. Since SHA-1 and SHA-2 are different, they hold similar algorithmic logic and eventually, specific hash lengths may be vulnerable to a similar collision attack. Since 2008, there have been public attacks on SHA-2 like SHA-1, and attacks against SHA-2 are getting severe over time. Some of the recent threats to SHA-2 were publicly declared in 2016. Likewise, it is expected that the existing hash will be attacked and become weaker over time. The federal authority of NIST selected SHA-3 as an improved hash standard, which is different from the SHA family and can be used when required. In 2010, Keccak Hash was mainly chosen as the only finalist. NIST released the standard in 2015, and SHA-3 became the certified standard. However, SHA-1 has promoted the entire world's development, and due to cryptography flaws, the significant shift towards SHA-2 took place in late 2016 and 2017. The deadline for the cut-off for obtaining SHA-1 certificates was on December 31, 2017. However, SHA-3 adoption is still in its infancy, and the reasons are listed:

- The prime reason is that implementation of SHA-3 is limited to the research domain only, and most of the existing software and hardware are yet not fully ready to support it. It also requires a customized code or program for each device to support it.
- If the recommended guidelines were to shift from SHA-1 to SHA-3, the hash vendors would have done this rather than moving to SHA-2 due to similar effort and cost.
- Most importantly, SHA-3 is a relatively new standardized hashing technique and was released when the SHA-2 migration schemes were still being explored.

- Another reason is that SHA-2 is an improved version of SHA-1, so it is not much vulnerable to collision attacks as SHA-1. Implementation of any version of the SHA-2 hashing algorithm is sufficient in the current scenario of today's digital world.

Many research studies also explored that SHA-3 is much slower than SHA-2. So, why shift over SHA-3, which is slower?

3.4 Reason for Shifting Towards SHA-3

The advancement of technology has laid out specific requirements that drive an upgrade to a new hashing technique to cope with the requirements needed in futuristic computing. SHA-3 is a robust technique compared to the existing hashing algorithms. In upcoming years, every system will undoubtedly shift to SHA-3. However, it depends on how the actual threats and security attacks on SHA-2 keep happening. In terms of software, the existing hashing schemes on windows machines are faster than SHA-3. Nonetheless, in terms of hardware, it easily defeated existing SHA-1 and SHA-2 hashing schemes. The cryptographic procedures are gradually controlled by hardware components in the future technology like an ecosystem of the Internet of Cyber-Physical Things (IoCPT). With the advancement of technologies, the CPU or processor is getting faster and faster. So, in most scenarios, the time required to hash will not cause much burden. Furthermore, few researchers [44], [45] have explored various ways to improve the speed of SHA-3 in software. In the context of the security strength of SHA-3 few significant points are highlighted as follows:

- SHA-3 provides a secure one-way hash function. More particularly, it is not susceptible to length expansion attacks. In this case, the input cannot be reconstructed using only the hash output, nor the input data can be altered by changing the hash value. Although SHA-3 offers the updated secure hash algorithm, the existing hashing algorithm SHA-2is still feasible for some applications.
- NIST still believes that only two identical hash functions (i.e., SHA-256 and SHA-512) of SHA-2 are secure.
- However, the newly released SHA-3 algorithm complements the existing SHA-2 while offering large varieties.

SHA-3 offers various functions with different digest bit lengths, including SHA-3-224, SHA-3-256, SHA-3-384, and SHA-3-512. It also offers two flexible output functions, SHAKE-128 and SHAKE-256, where 128 and 256 extensions are security robustness factors, which can be utilized to attain global and randomized hashing, hash-based message authentication code and even for performing stream encryption.

4.0 STATISTICAL ANALYSIS

This section discusses the statistical analysis being carried out over different variants of the SHA family. An open-source platform of Java is considered for experimenting with a standard 64-bit Windows machine. The proposed analysis considers Java Cryptography API to generate the hashes [46], assuming a sample binary string of ten thousand random messages. The digest size for input and output is retained for the more straightforward analysis. The statistical analysis is carried out for SHA-1 with 160 bits and SHA-2 and SHA-3 with 512 bits. Although there are various SHA-2 and SHA-3, the proposed analysis chooses a 512-bit variant of both. The statistical analysis in the proposed study is carried out using the series test, bits probability test, and Hamming distance test.

4.1 Series Test

The proposed system carries out a non-parametric test that considers random samples from multiple populations associated with the cumulative distribution of continuous data. The prime idea of this test is to measure the state of the randomness of the hashes with clear insight towards studying the dependencies of each hash function. The mathematical expression of assessing test statistics $\emptyset$ is as follows:

$$\emptyset = \frac{\Delta\psi}{\sigma} \tag{2}$$

In expression (2), $\Delta\psi$ is evaluated by the difference between ψ and ψ_1, which represents series cardinality depicted by one hash and anticipated cardinality of series. The variable σ represents the standard deviation. Further, the computation of ψ_1 depends upon two parameters of the cardinality of hashes, i.e., favourable cardinality c_f and total cardinality c_t, while their empirical relationship is as follows:

$$\psi_1 = \frac{2c_f}{c_t} + 1 \tag{3}$$

In expression (3), c_f and c_t are computed as $(c_0 * c)$ and $(c_0 + c)$, where c_0 and c represent the cardinality of sub-sequences which has all the 0 and 1, respectively. Considering the significance level of 5%, the proposed system carries out the series test. It will mean that the hypothesis for random creation of hash will be proven effective if the absolute value of test statistics $\emptyset$ is more than test statistics corresponding to 0.975, which will be 1.96.

Table 3: Accomplished Numerical Outcome of Series Test

Item	Hash Function		
	SHA-1	SHA-2	SHA-3
Maximum	5.34	5.15	5.41
Minimum	0	0	0
Average	0.81	0.91	0.81
Standard Deviation	±0.71	±0.72	±0.61

From the above numerical outcomes (Table 3), SHA-3 is the lowest value of standard deviation that is statistically significant to exhibit the best test outcome compared to the SHA-1 and SHA-2 family of hashes.

4.2 Bits Probability Test

This is the second statistical test carried out in the proposed system to assess the predictability of the bits present in the MD. As a result, the study computes the probability of one second for every position of the message bit. Accordingly, the ideal condition will confirm 0 in 50% probability while 1 in another 50% probability. Mathematically, it can be represented as follows:

$$Prob_i(i) = \frac{\sum_{j=1}^{max} h[j][i]}{max} \tag{4}$$

In the above expression (4), the computation of probability $Prob$ is carried out by considering the maximum position of bit max and a maximum length of hash. The expression also represents h as the hash values of the table associated with yielded MD. The assessment considers the max value to be 10000 in expression (3). The proposed analysis is carried out by considering test statistics $|\emptyset|$ discussed in expression (4) in the following sub-section. With an anticipated outcome of 50% probability, the assessment is regarded as a pass if the value $|\emptyset|$ is found to be less than 1.96. The numerical outcome is shown in Table 4.

Table 4: Accomplished Numerical Outcome of Bit Prediction

Item	Probability		
	SHA-1	SHA-2	SHA-3
Maximum	52.46	52.57	52.85
Minimum	49.31	49.51	49.87
Average	51.15	49.01	50.01
Standard Deviation	±0.53	±0.54	±0.52

A closer look at the above numerically tabulated value shows the average value in the proximity of 50% with a reduced standard deviation. It can also be noted that the performance of both SHA-1 and SHA-2 are nearly equivalent, where the analysis found 510 bits to possess a probability value that is different in comparison to 50%. While, the computation of absolute test statistics $|\emptyset|$ for SHA-1 and SHA-2 is found to be 1.05 and 0.82, respectively, representing the higher predictability property of both hash functions. On the other hand, the numerical outcome of SHA-2 is also found slightly to be equivalent to SHA-3. It was found that 507 bits out of 512 bits do not meet the condition of 50% probability with a minor difference. The analyzed test statistics value of $|\emptyset|$ is 0.328, demonstrating a lower prediction probability score than SHA-1 and SHA-2. Based on this test, it can be said that the proposed outcome exhibits lower predictability for SHA-3 than SHA-1 and SHA-2.

4.3 Hamming Distance Test

This is the third statistical analysis in the proposed evaluation to identify the significant impact of minor alteration in data towards outcome hash files. This assessment form involves applying a t-student test to compute the test statistics $|\emptyset|$. This test constructs a hypothesis that the hash function can successfully pass through if the numerical score of the test statistic resides between 0 and 1.96 of the confidence intervals. The anticipated outcome of the test statistics is equivalent to half the hash size with 5% of the significance level. The mathematical formulation for this assessment of test statistics $|\emptyset|$ can be represented as shown in expression (5):

$$|\emptyset| = |\frac{v_{av}-v_{ex}}{\sigma}\sqrt{s_{size}}| \tag{5}$$

The prime purpose of this part of the analysis is to compute the Hamming distance between the hash. The operation carried out in this analysis is: considering X_1 and X_2 are primary and secondary bits of information, then the length of X_1 is equivalent to the size of X_2. Therefore, Hamming distance computation is carried out as $X_3 = X_1 \oplus X_2$. This distance represents the cardinality of position with discretely different values for both X_1 and X_2. The numerical outcomes of Hamming distance are shown in Table 5.

Table 5: Accomplished Numerical Outcome of Series Test

Item	Hash Function		
	SHA-1	SHA-2	SHA-3
Maximum	5.34	5.15	5.41
Minimum	0	0	0
Average	0.81	0.91	0.81
Standard Deviation	±0.71	±0.72	±0.61

From the numerical outcomes of Hamming distance in Table 5, it can be seen that absolute test statistics $|\emptyset|$ for SHA-1 is about 1.28, which will mean that a minor alteration in the input data will affect 50% of the hashes of SHA-1. Regarding SHA-2, the hash variation is sometimes closer to and lower than 50%, while the average test statistic $|\emptyset|$ is about 0.159. This outcome shows SHA-2 to perform better than SHA-1. From the viewpoint of SHA-3, in the majority of the cases, the critical values of test statistics are found to be 0.44, which exhibits SHA-2 is still the best outcome. At the same time, SHA-3 and SHA-1 occupy the third and second positions, respectively.

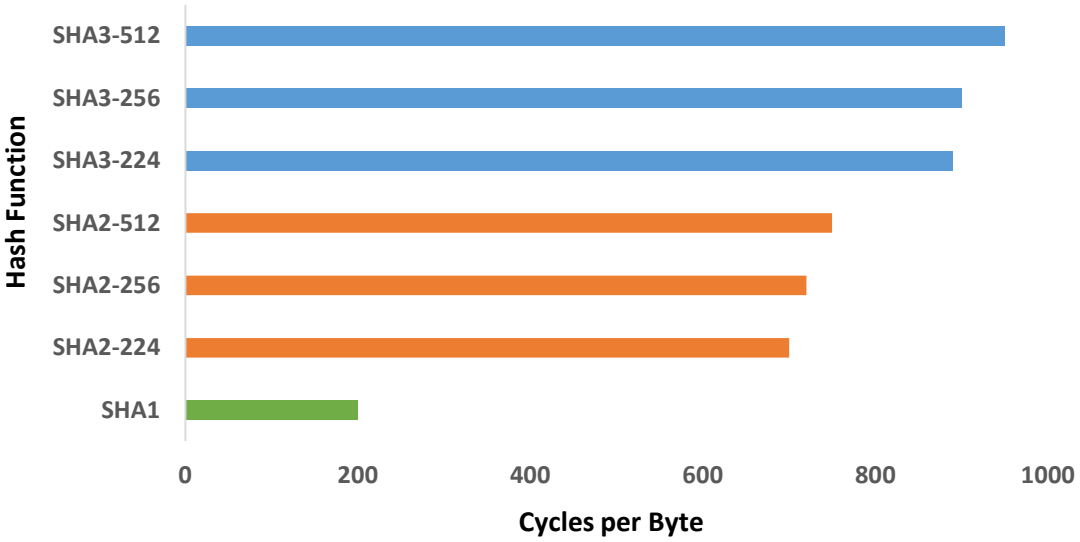
5.0 PERFORMANCE ANALYSIS

The performance analysis of the SHA family is carried out considering the number of cycles utilized for data (byte) processing. Unlike conventional analysis mechanisms using data transmitted per unit time, the proposed analysis considers the processor's clock cycles to perform data processing. To compute the number of cycles utilized in processing one unit of data associated with the encrypted hash function, the amount of the data processed is divided by the cycles it consumes to process the data. It is to be noted that this performance parameter could significantly differ from one to another encryption option. The prime justification behind adopting this performance parameter is that it offers the possibility to compute total duration, which will mean that the processor with maximal frequency will exhibit better performance. Furthermore, this performance parameter will directly indicate the effective processing capability of all kinds of cryptographic operations.

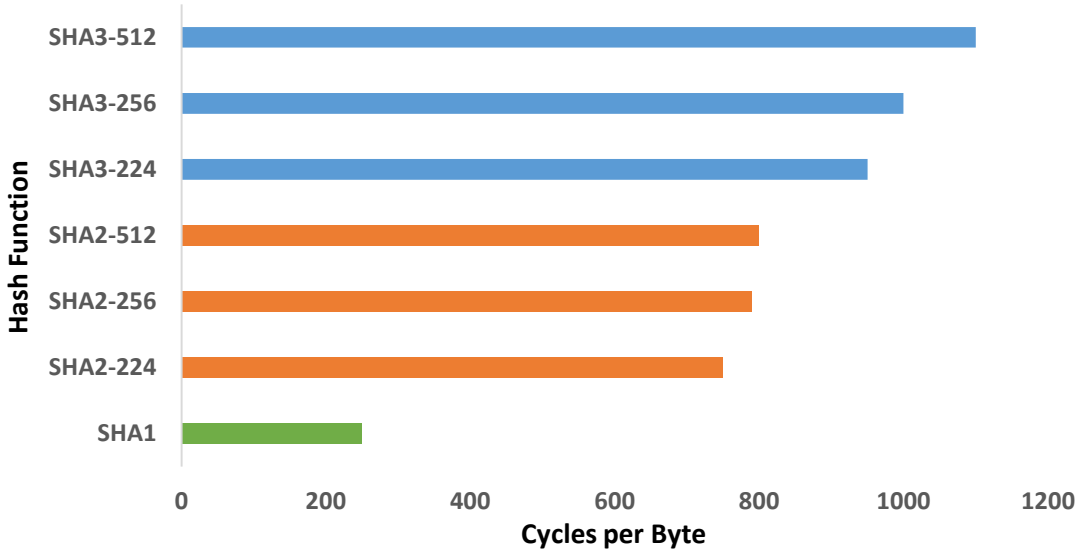
The computation of the cycles per data is carried out as follows:

$$cyclesperdata = \frac{(D_h * \lambda)}{len} \tag{6}$$

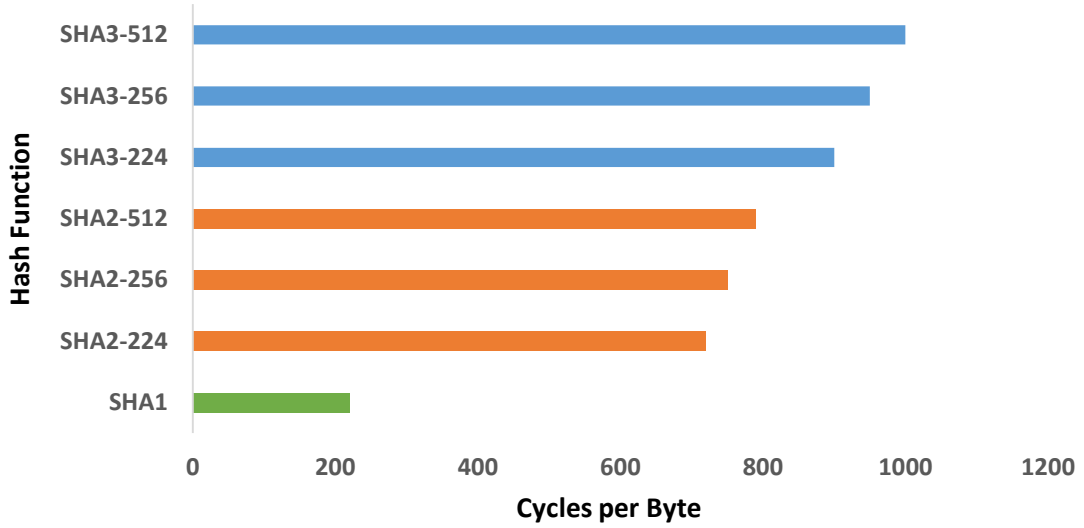
Expression (6) shows that cycles per data depend on the duration of hashing operation D_h, CPU frequency λ, and length of input message len. To carry out this performance analysis, the proposed study considers cycles per data (or byte) as the prime observational values in the presence of 1 kB, 1 MB, and 64 MB input message size in a typical 64-bit windows environment. This evaluation method will give a complete idea of the scalability performance of the SHA approach over a similar test environment.


SHA3-512
SHA3-256
SHA3-224
SHA2-512
SHA2-256
SHA2-224
SHA1
Hash Function
0
200
400
600
800
1000
Cycles per Byte

(a)


SHA3-512
SHA3-256
SHA3-224
SHA2-512
SHA2-256
SHA2-224
SHA1
Hash Function
0
200
400
600
800
1000
1200
Cycles per Byte

(b)



(c)

Fig. 8: Performance Analysis for a) 1 KB, b) 1 MB, and c) 64 MB input data

The outcome exhibited in Fig. 8 highlights three different test cases to assess the clock usage per byte involved for three different variants of the hash function. Fig. 8(a) shows that SHA-1 has the lower occupation of around 200 cycles per byte, which is the best result compared to the other two variants of the hash function, i.e., SHA-2 and SHA-3. A closer look into SHA-2 shows no significant increase in cycles ranging between 700-800 cycles per byte. At the same time, SHA-3 and its three variants range between 890-1100 cycles per byte. A similar trend is observed in Fig. 8(b) and Fig. 8(c). Another observation among the three graphical outcomes shows an increase in cycles per byte with the input message size. Despite the reduced occupation of cycles per byte by SHA-1, it cannot be considered adequate concerning its security operation. It is known that SHA-1 is anticipated to be operational in no two segments that operate through the internal process, which is again expected to be equivalent to the same hash. It is also known that the hash of SHA-1 usually is 160 bits long (i.e., a string of 160 zeroes and ones), referring to the presence of 2160 or 1.4 quindecillion possibilities of hash. This is significantly less when compared with SHA-2 variants. Another significant observation is that theoretically, SHA-2 offers better performance than SHA-3 variants, especially regarding SHA-2-224 and SHA-2-512. SHA-2 variants utilize a structure of Davies-Meyer considering a block cipher that is constructed from MD4. On the other hand, SHA-3 variants make use of sponge structure considering the permutation of Keccak. This makes the complete internal structure different for both the variants of SHA-2 and SHA-3.

There are no concrete criteria to confirm that anyone has more potential based on the outcome. SHA-1 possesses structural weakness, making it vulnerable to attacks like brute force. On the other hand, SHA-2 and SHA-3 are the same and cannot be considered to possess structural weaknesses. On the contrary, SHA-3 is slower than SHA-2, as exhibited by more cycles per byte in Fig. 8(b) and Fig. 8(c). Hence, from performance analysis based on cycles per byte, SHA-2 confirms to offer better performance. However, from a security viewpoint, there are few potential benefits of SHA-3 compared to SHA-2. The prime contribution of SHA-3 is its Keccak sponge that can be utilized both in the form of Media Access Control (MAC) and hash, unlike SHA-2, which also results in an increase of cycles per byte in the presented outcome of Fig. 8. It is used as a function that can derive the secret key cost-effectively. Although SHA-3 has more cycle per byte inclusion and will demand slightly more resources, they are still cost-effective security solutions. This outcome shows the higher applicability of SHA-3 on Internet of Things (IoT) systems, which requires using the low-powered device with cost-effectiveness. As the construction of SHA-3 completely differs from SHA-2 variants, it is unlikely to apply SHA-3 in case of new intruder breaks in SHA-2 or vice-versa. Therefore, both SHA-2 and SHA-3 must be emphasized equally for making a more straightforward transition to SHA-3 until SHA-3 is proven to be 100% effective from a performance and security viewpoint.

6.0 EXTENSIVE FAULT ANALYSIS

An extensive fault analysis using differential and Algebraic Fault Analysis (AFA) strategies to evaluate the strength of all three variants of the SHA algorithm was carried out. This part of the analysis considered the message size for

SHA-2 and SHA-3 limited to 224 and 256 only as there were no significant differences in the outcome achieved in 256 and 512 message sizes for two variants of SHA, i.e., SHA-2 and SHA-3. The prime objective of the Differential Fault Analysis (DFA) was to retrieve the information of an inner state by using the difference in the correlation among the defective resultants and all intermediate parameters in contrast to the appropriate resultant. DFA was initially used to analyze block ciphers' strength, DES algorithm, hash functions and stream ciphers [47]. On the other hand, a different variant called Algebraic Fault Analysis (AFA) integrates algebraic crypto analysis with fault injection; and an algebraic expression is used over a finite field geometry to translate faults and cryptographic function. Technology of Satisfiability (SAT) Solver or Satisfiability Modulo Theories (SMT) is often used for recovering the message or secret key in such a mechanism [48].

It has been noted that analysis is more effective when carried out with AFA than DFA as the solver's complete propagation of fault reduces the dependency on human intervention to introduce the intrusion. Moreover, the application of AFA was found to automate the DFA mechanism toward hash function, stream ciphers and block ciphers [47]. The recovery problem associated with an internal state has been investigated in the proposed system concerning SHA-1, SHA-2 (224, 256), and SHA-3 (224, 256). This was carried out using the boolean expression for the operation, followed by using SAT solver to explore the solution for all parameters connected with confidential information. It was carried out by constructing an equation set for all hash functions associated with faulty and appropriate execution. The expression for a proper hash H considering the input of L_i^{22} as:

$$H = A.B \tag{7}$$

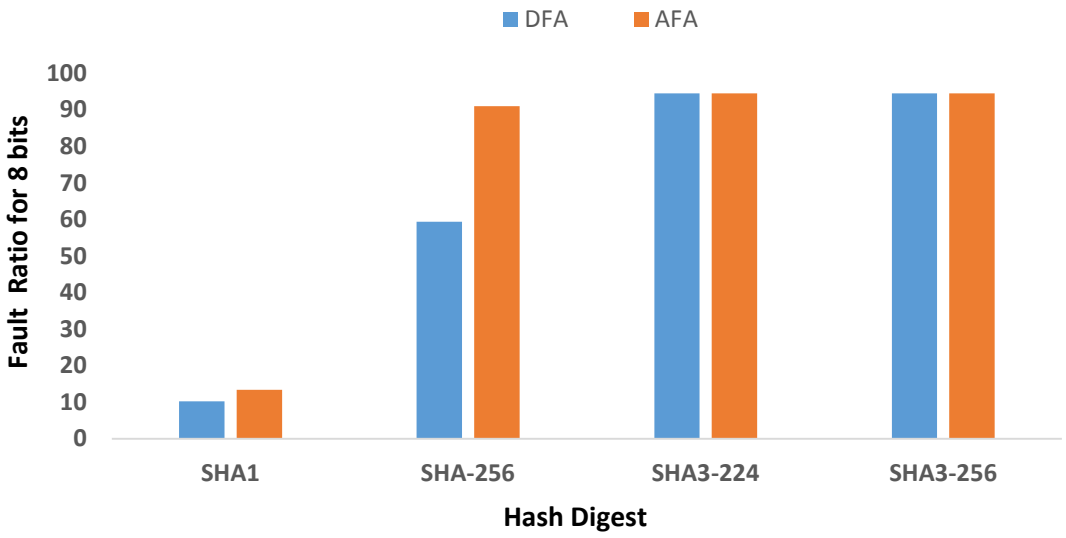
In expression (7), the variable $A = f(\psi 23, B, S, R, L)$ and $B = f(\psi 22, B, S, R, L(L_i^{22}))$, where $\psi 23/\ \psi 22$ represents binary XOR operation, B represents a binary operation in rows over state bits, S represents in-slice permutation over state bits, R represents rotation over state bits L represents linear operation with all input bits and single output bits. The above expression is slightly modified to include a fault in the form of $\Delta L\_i\hat{}22$ to generate a fault hash of:

$$H1 = A.B1 \tag{8}$$

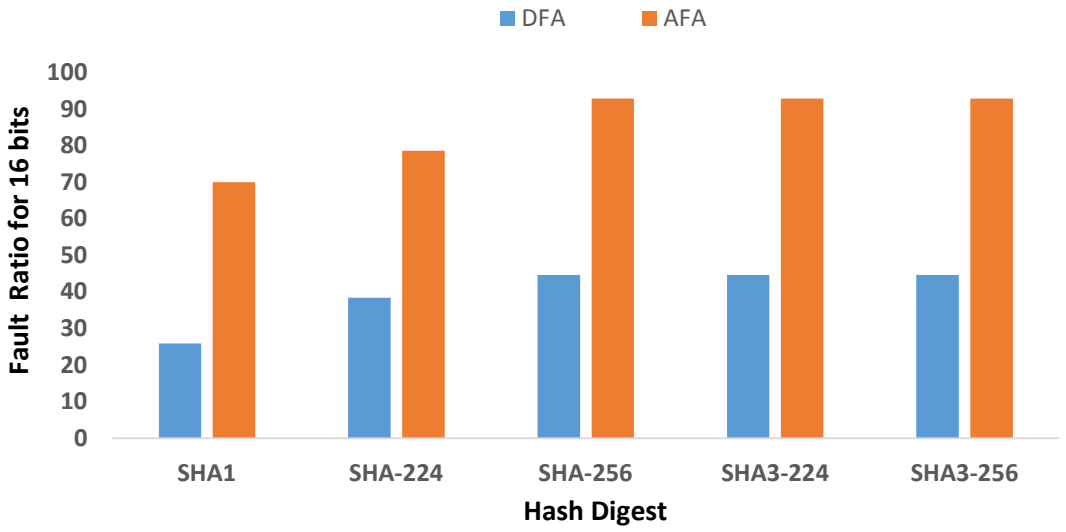
In expression (8), $B1$ represents $B1 = f(\psi 22, B, S, R, L(L\_i\hat{}22 \oplus \Delta L\_i\hat{}22))$. Considering the individual cases of all the variants of a hash function, the value of H and $H1$ in expressions (1) and (2) can be x-bits. A closer look into this strategy will showcase that the value of $\Delta B\_i\hat{}23$ can be extracted using H and $H1$ differential while the same attacker can use $B\_i\hat{}23$ to launch an attack.

$$Bo(x, y, z) = Bi(x, y, z) \oplus Bi(x + 2, y, z) \oplus Bi(x + 1, y, z).Bi(x + 2, y, z) \tag{9}$$

However, the expression (9) mentioned above differs for all the three hash functions, but it will perform faster for AFA, recovering $B\_i\hat{}22$ bits of data. It should be carefully noted that there is a significant level of inter-dependencies towards all the internal states with reversible functions. It will mean that a compromising attempt toward the hash algorithm can be carried out by extracting only one internal state. The solution to the proposed scheme of fault analysis will be just one uniquely recovered bit of information. Therefore, there is a need to explore $B\_i\hat{}22$ bits of information in the influence of fault injection. The SAT solver is executed to explore the initial rounds of solution and then reduce them to non-repeating bits.



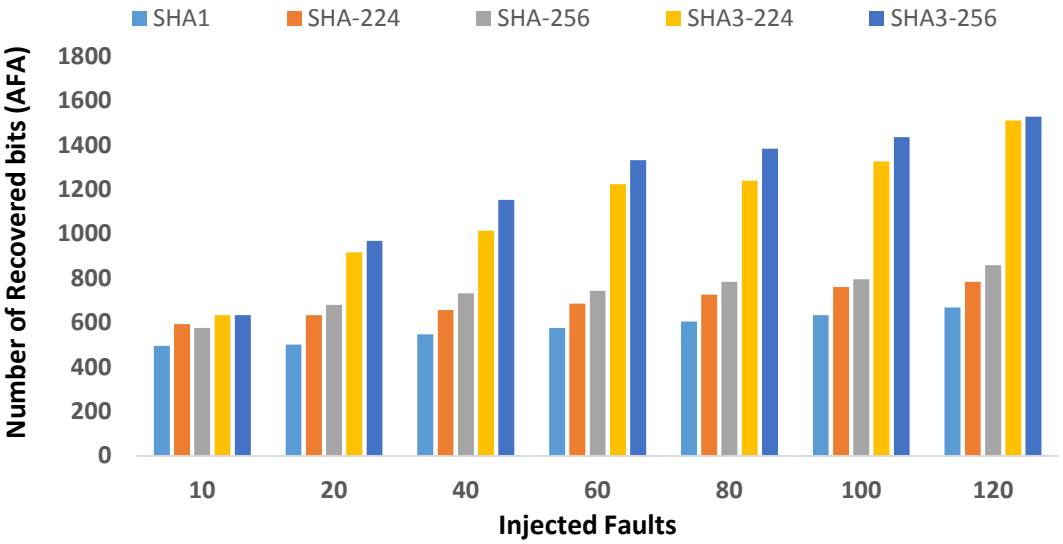
(a)



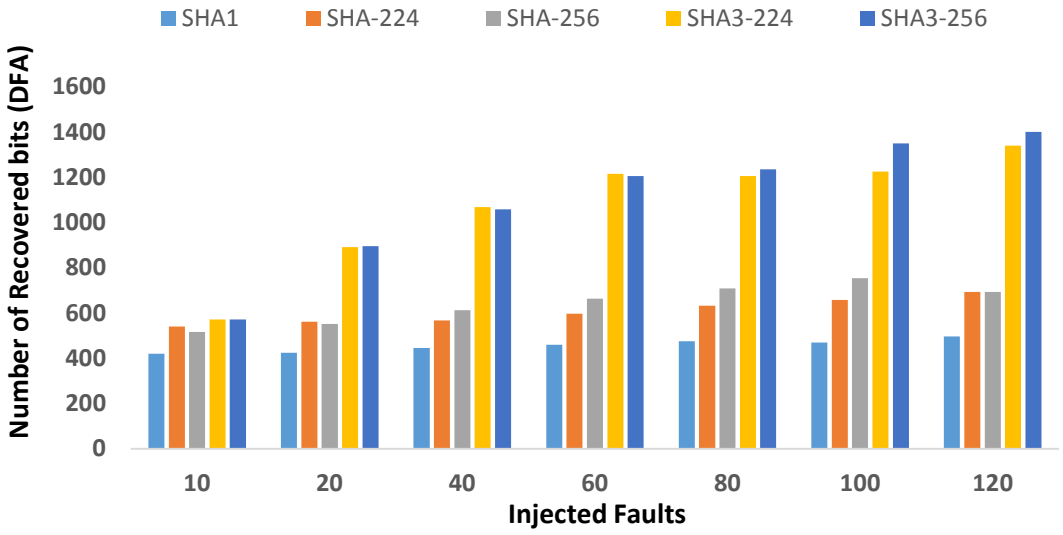
(b)

Fig. 9: Comparative Analysis of Fault Ratio (a) For 8 Bits and (b) For 16 Bits

The outcome shown in Fig. 9 highlights that both the variants of SHA-3 offer a higher value of fault ratio than the two variants of SHA-2 and SHA-1 when assessed using 8 and 16-bit values, respectively. An intruder can recognize the injected fault associated with a particular injection. The ratio of effective fault can be computed by considering a cumulative number of feasible faults for a specific number of positions with all possible fault values. This outcome showcases that adopting the SHA-3 variant for higher message size results in higher recovery of bits (Fig. 10) irrespective of AFA and DFA consideration. The outcome eventually infers that SHA-3 variants are always practical to adopt while dealing with a new generation of attacks; however, SHA-2 variants will also be capable enough to deal with previously reported attacks in literature.

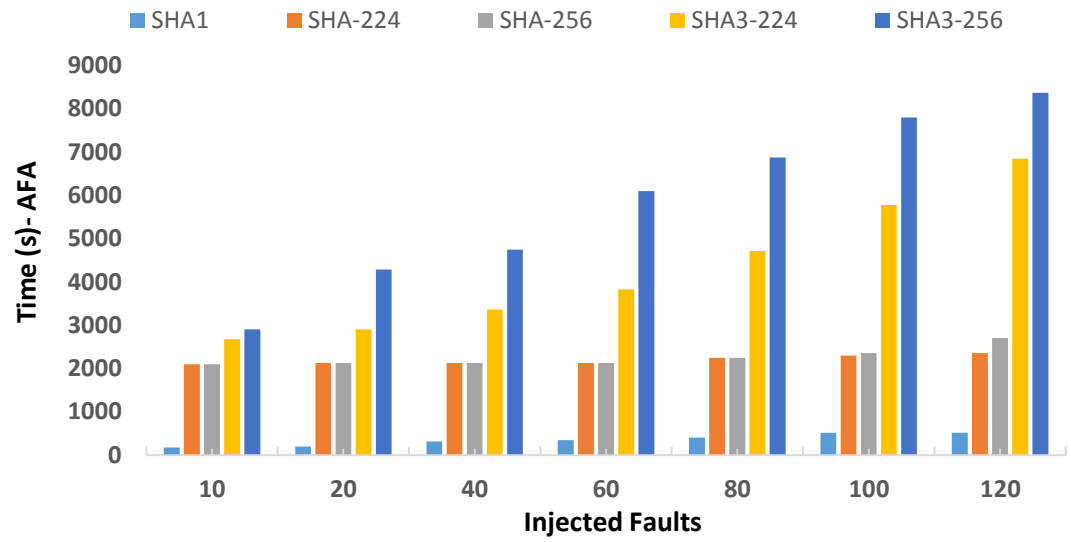


(a)

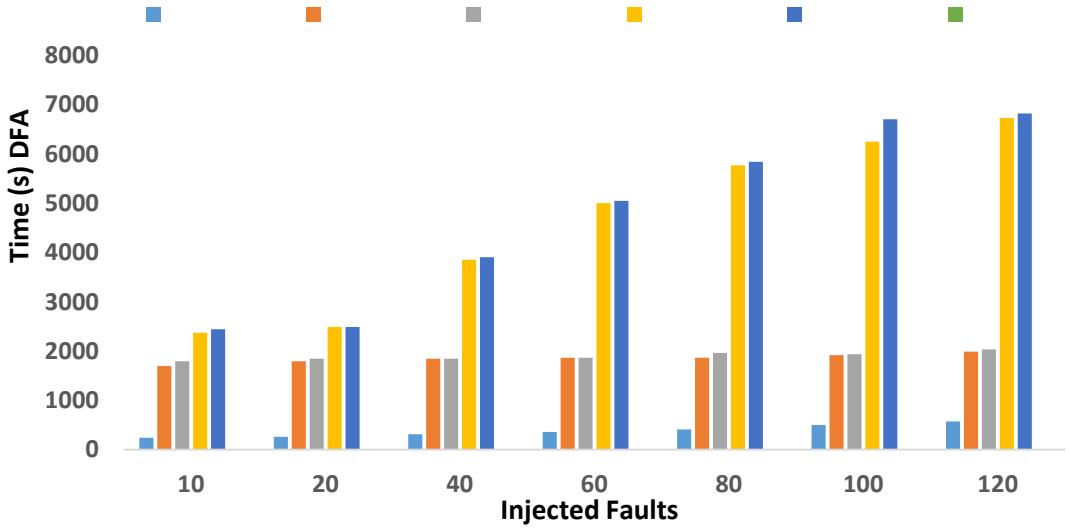


(b)

Fig. 10: Comparative Analysis of Recovered Bits: a) For AFA and b) For DFA



(a)



(b)

Fig. 11: Comparative Analysis of Processing Time (a) For AFA and (b) For DFA

The next part of the analysis investigates the recovered bits (Fig. 10). It is feasible for attackers to inject a different number of faults associated with the same input and execute all hash variant functions targeting the retrieval of the internal state of $B\_i\hat{}22$. The result is presented for the recovery process of SHA-1, SHA-224, SHA-256, SHA-3-224, and SHA-3-256 to see the higher capability of AFA compared to DFA with more $B\_i\hat{}22$ bits. The outcome shows that SHA-3 offers lesser dependencies of less fault injection than other SHAs to recover the total value of the bit state. At the same time, analysis is carried out regarding the time required to perform cryptanalysis. (Fig. 11) showcases no significant difference between AFA and DFA performance for SHA-3 and SHA-2, but SHA-3 offers more time than SHA-2 variants. This is because the SAT solver required more time to recognize the faults, followed by recovering the internal states of the bits. Hence, the proposed investigation shows that SHA-3 offers better security features than other SHA variants.

7.0 STUDY OUTCOME

After performing an extensive assessment of the SHA family, the following learning outcomes have been drawn:

- The statistical analysis exhibits that SHA-3 has the lowest randomness of hashes than SHA-2 in the series test. Again, SHA-3 performance was found to have a lower predictability score, as seen from the bit probability test. This signifies that even with a less state of randomness, SHA-3 is hard to predict for a given number of bits present in the MD. This outcome makes SHA-3 more robust and resilient from unknown attackers. The Hamming distance test shows better results for SHA-2 compared to SHA-3. However, the sponge construction of SHA-3 as an internal structure will offer more security strength for novice forms of attackers. Hence, SHA-3 is the best complimentary solution on top of the SHA-2 approach.
- The performance analysis exhibits that all the variants of SHA-3 offer increased clock per byte in contrast to SHA-1 and SHA-2. This outcome was justified by the internal structure of SHA-3, making it more secure due to the presence of MAC and the hash, which is not the case with SHA-2. Irrespective of more cycles per byte, SHA-3 is still the best option for offering authentication and data integrity. Due to its flexible structure, SHA-3 provides equivalent performance to non-anonymity, non-repudiation and privacy from maximum attacker forms.
- The extensive fault analysis exhibits a higher recovery of bits for SHA-3, while the processing time for the SHA-3 is nearly the same for all of its respective variants. Although the processing time is slightly higher with increasing injected faults than SHA-2 and SHA-1, SHA-3 is still the preferred algorithm for resisting higher-end intruders with low faults in the cumulative outcome.

Based on the above outcomes, it can be said that SHA-3 should be used in a specific environment where the attacker's strategy is less known. If the attacker strategy is well defined, SHA-2 will be quite enough to mitigate such attacks. SHA-1 is not recommended due to the limitation observed from the analysis discussed in prior sections. Hence, the outcome suggests wiser use of SHA-3 in a complex attack environment, whereas SHA-2 can still be used for normal to medium vulnerability.

8.0 CONCLUSION

Hash functions play a vital role in network and communication security. The study and its contribution are manifold:

- The analysis offers a justified outcome towards the usage of different variants of SHA, which has not been reported in existing research publications,
- The implementation considers a uniform test environment subjected to SHA family using three different analysis mechanisms to offer validated outcome,
- The outcome removes the myth that one variant of SHA is more effective than its counterparts instead, it concludes that
  - SHA-1 is less efficient in the majority of performance attribute while
  - SHA-2 and SHA-3 bears potential to resist the majority of the threats,
- The study outcome continues to offer proof that SHA-3 is in a better position to mitigate the new definition of the attacker. At the same time, SHA-2 will be suitable enough to deal with the existing version of the known attacker,
- The study also offers information that although SHA-3 has slightly more processing time, it can still be considered to look into its potential in other performance metrics.

The main problem of any hashing algorithm is to check the integrity and authenticity of the data transmitted between two communicating nodes. This paper discussed the main highlights of cryptographic hash functions, the widespread use of various well-known hash functions and associated attacks. The study also carried out a comparative analysis of different hash algorithms and analyzed the trend of progress in this field. Studies on hash functions performed by other researchers were also discussed. However, most of them were limited to theoretical implementation and not tested against collision attacks. Based on the analytical findings, it has been shown that the current status and trend toward adopting SHA-3 were efficient, safe and capable of meeting the requirements of future network applications. Certain pitfalls were observed in the usage of the SHA-2 and SHA-3 families, which are associated with partial breakdown connected to bitwise operators. The beneficial point of SHA-3 is associated with its capability of faster response time and considering the maximum size of data for performing encryption. Hence, future works should address this problem by reducing clock cycles and quicker response time.

9.0 ACKNOWLEDGEMENT

This research was partially funded by IIUM-UMP-UiTM Sustainable Research Collaboration Grant 2020 (SRCG) under Grant ID: SRCG20-003-0003 and Fundamental Research Grant Scheme (FRGS) under Grant ID FRGS19-068-0676. The authors express their personal appreciation for the effort of Ms Gousia Nissar and Ms Manasha Saqib in proofreading, editing and formatting the paper.

REFERENCES

[1] T. Wang, M. Bhuiyan, G. Wang, L. Qi, J. Wu and T. Hayajneh, "Preserving Balance Between Privacy and Data Integrity in Edge-Assisted Internet of Things", *IEEE Internet of Things Journal*, Vol. 7, No. 4, 2020, pp. 2679-2689, doi: 10.1109/jiot.2019.2951687.

[2] D. Chen, P. Bovornkeeratiroj, D. Irwin and P. Shenoy, "Private Memoirs of IoT Devices: Safeguarding User Privacy in the IoT Era", in 2018 *IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, 2018, pp. 1327-1336, doi: 10.1109/ICDCS.2018.00133.

[3] Z. A. Solangi, Y. A. Solangi, S. Chandio, M. bt. S. Abd. Aziz, M. S. bin Hamzah and A. Shah, "The Future of Data Privacy and Security Concerns in Internet of Things", in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, Bangkok, Thailand, 2018, pp. 1-4, doi: 10.1109/ICIRD.2018.8376320.

[4] I. Ochôa, L. Calbusch, K. Viecelli, J. de Paz, V. Leithardt and C. Zeferino, "Privacy in the Internet of Things: A Study to Protect User's Data in LPR Systems Using Blockchain", in *2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton*, NB, Canada, 2019, pp. 1-5, doi: 10.1109/PST47121.2019.8949076.

[5] M. Khari, A. K. Garg, A. H. Gandomi, R. Gupta, R. Patan and B. Balusamy, "Securing Data in Internet of Things (IoT) Using Cryptography and Steganography Techniques", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 50, No. 1, 2020, pp. 73-80, doi: 10.1109/TSMC.2019.2903785.

[6] H. Al Hamid, S. Rahman, M. Hossain, A. Almogren and A. Alamri, "A Security Model for Preserving the Privacy of Medical Big Data in a Healthcare Cloud Using a Fog Computing Facility with Pairing-Based Cryptography", *IEEE Access*, Vol. 5, 2017, pp. 22313-22328, doi: 10.1109/access.2017.2757844.

[7] N. Sharma, H. Parveen Sultana, R. Singh and S. Patil, "Secure Hash Authentication in IoT based Applications", *Procedia Computer Science*, Vol. 165, 2019, pp. 328-335, doi: 10.1016/j.procs.2020.01.042.

[8] S. Suhail, R. Hussain, A. Khan and C. S. Hong, "On the Role of Hash-Based Signatures in Quantum-Safe Internet of Things: Current Solutions and Future Directions", *IEEE Internet of Things Journal*, Vol. 8, No. 1, 2021, pp. 1-17, doi: 10.1109/JIOT.2020.3013019.

[9] K. Saravanan and A. Senthilkumar, "Theoretical Survey on Secure Hash Functions and Issues", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2, No. 10, 2013, pp. 1150-1153.

[10] A. Jurcut, T. Niculcea, P. Ranaweera and N. Le-Khac, "Security Considerations for Internet of Things: A Survey", *SN Computer Science*, Vol. 1, No. 4, 2020, pp. 1-19, doi: 10.1007/s42979-020-00201-3.

[11] H. Huang, Q. Huang, F. Xiao, W. Wang, Q. Li and T. Dai, "An Improved Broadcast Authentication Protocol for Wireless Sensor Networks Based on the Self-Reinitializable Hash Chains", *Security and Communication Networks*, Vol. 2020, 2020, pp. 1-17, doi: 10.1155/2020/8897282.

[12] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, "The First Collision for Full SHA-1", in *Annual International Cryptology Conference*, Santa Barbara, USA, 2017, pp. 570-596, doi: 10.1007/978-3-319-63688-7_19.

[13] R. Martino and A. Cilardo, "SHA2 Acceleration Meeting the Needs of Emerging Applications: A Comparative Survey", *IEEE Access*, Vol. 8, 2020, pp. 28415-28436, doi: 10.1109/access.2019.2920089.

[14] A. Mohammed Ali and A. Kadhim Farhan, "A Novel Improvement with an Effective Expansion to Enhance the MD5 Hash Function for Verification of a Secure E-Document", *IEEE Access*, Vol. 8, 2020, pp. 80290-80304, doi: 10.1109/ACCESS.2020.2989050.

[15] H. Liu, A. Kadir and J. Liu, "Keyed Hash Function Using Hyper Chaotic System with Time-Varying Parameters Perturbation", *IEEE Access*, Vol. 7, 2019, pp. 37211-37219, doi: 10.1109/ACCESS.2019.2896661.

[16] M. Rathor and A. Sengupta, "IP Core Steganography Using Switch Based Key-Driven Hash-Chaining and Encoding for Securing DSP Kernels Used in CE Systems", *IEEE Transactions on Consumer Electronics*, Vol. 66, No. 3, 2020, pp. 251-260, doi: 10.1109/TCE.2020.3006050.

[17] J. Ouyang, X. Zhang and X. Wen, "Robust Hashing Based on Quaternion Gyrator Transform for Image Authentication", *IEEE Access*, Vol. 8, 2020, pp. 220585-220594, doi: 10.1109/ACCESS.2020.3043111.

[18] Y. Zhou, B. Yang, Z. Xia, Y. Mu and T. Wang, "Anonymous and Updatable Identity-Based Hash Proof System", *IEEE Systems Journal*, Vol. 13, No. 3, 2019, pp. 2818-2829, doi: 10.1109/JS2018.2878215.

[19] D. I. Nassr, "Secure Hash Algorithm-2 formed on DNA", *Journal of the Egyptian Mathematical Society*, Article No. 24, 2019, pp. 1-20, doi: 10.1186/s42787-019-0037-6.

[20] Y. Lee, S. Rathore, J. H. Park, and J. H. Park, "A Blockchain-Based Smart Home Gateway Architecture for Preventing Data Forgery", *Human-centric Computing and Information Science*, Vol. 10, No. 1, 2020, pp. 1-14, doi: 10.1186/s13673-020-0214-5.

[21] M. A. Rezazadeh Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "Authentication Strategies in Vehicular Communications: A Taxonomy and Framework", *EURASIP Journal on Wireless Communication and Networking*, Vol. 2021, No. 1, 2021, pp. 1-50, doi: 10.1186/s13638-021-01968-6.

[22] R. Martino and A. Cilardo, "SHA2 Acceleration Meeting the Needs of Emerging Applications: A Comparative Survey", *IEEE Access*, Vol. 8, 2020, pp. 28415-28436, doi: 10.1109/ACCESS.2020.2972265.

[23] I. E. Salem, A. M. Salman and M. M. Mijwil, "A Survey: Cryptographic Hash Functions for Digital Stamping", *Journal of Southwest Jiaotong University*, Vol. 54, No. 6, 2019, pp. 1-11, doi: 10.35741/issn.0258-2724.54.6.2.

[24] Jianhua Mo, Xiawen Xiao, Meixia Tao and Nanrun Zhou, "Hash Function Mapping Design Utilizing Probability Distribution for Preimage Resistance", in *2012 IEEE Global Communications Conference (GLOBECOM)*, Anaheim, CA, 2012, pp. 862-867, doi: 10.1109/GLOCOM.2012.6503221.

[25] B. Preneel, "Second Preimage Resistance", *Encyclopedia of Cryptography and Security*, Boston, MA, Springer, 2005, doi: 10.1007/978-3-540-25937-4_24.

[26] A. Maetouq and S. M. Daud, "HMNT: Hash Function Based on New Mersenne Number Transform", *IEEE Access*, Vol. 8, 2020, pp. 80395-80407, doi: 10.1109/ACCESS.2020.2989820.

[27] W. Jing, D. Zhang and H. Song, "An Application of Ternary Hash Retrieval Method for Remote Sensing Images in Panoramic Video", *IEEE Access*, Vol. 8, 2020, pp. 140822-140830, doi: 10.1109/ACCESS.2020.3006103.

[28] H. Cui, L. Zhu, J. Li, Y. Yang and L. Nie, "Scalable Deep Hashing for Large-Scale Social Image Retrieval", *IEEE Transactions on Image Processing*, Vol. 29, 2020, pp. 1271-1284, doi: 10.1109/TIP.2019.2940693.

[29] Y. Zheng, Y. Cao and C. H. Chang, "UDhashing: Physical Unclonable Function-Based User-Device Hash for Endpoint Authentication", *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 12, 2019, pp. 9559-9570, doi: 10.1109/TIE.2019.2893831.

[30] A. Biswas, A. Majumdar, S. Nath, A. Dutta and K. L. Baishnab, "LRBC: A Lightweight Block Cipher Design for Resource Constrained IoT Devices", *Journal of Ambient Intelligence and Humanized Computing*, 2020, pp. 1-15, doi: 10.1007/s12652-020-01694-9.

[31] P. Chanal and M. Kakkasageri, "Security and Privacy in IoT: A Survey", *Wireless Personal Communications*, Vol. 115, No. 2, 2020, pp. 1667-1693, doi: 10.1007/s11277-020-07649-9.

[32] E. Molina and E. Jacob, "Software-Defined Networking in Cyber-Physical System: A Survey", *Computers & Electrical Engineering*, Vol. 66, 2018, pp. 407-419, doi: 10.1016/j.compeleceng.2017.05.013.

[33] I. Graja, S. Kallel, N. Guermouche, S. Cheikhrouhou and A. Hadj Kacem, "A Comprehensive Survey on Modeling of Cyber‐Physical Systems", *Concurrency and Computation: Practice and Experience*, Vol. 32, No. 5, 2018, pp. 1-18, doi: 10.1002/cpe.4850.

[34] B. U. I. Khan, R. F. Olanrewaju, F. Anwar and M. Yaacob, "Offline OTP Based Solution for Secure Internet Banking Access", in *2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, Langkawi, Malaysia, 2018, pp. 167-172, doi: 10.1109/IC3e.2018.8632643.

[35] B. U. I. Khan, R. F. Olanrewaju, F. Anwar, R. N. Mir and A. Najeeb, "A Critical Insight into the Effectiveness of Research Methods Evolved to Secure IoT Ecosystem", *International Journal of Information and Computer Security*, Vol. 11, No. 45, 2019, pp. 332-354, doi: 10.1504/ijics.2019.10023470.

[36] C. Jin, "Cryptographic Solutions for Cyber-Physical System Security", *Doctoral Dissertation*, University of Connecticut, Storrs, 2019.

[37] A. Tawalbeh and H. Tawalbeh, "Lightweight Crypto and Security", *Security and Privacy in Cyber-Physical Systems*, John Wiley & Sons, 2017, pp. 243-261, doi: 10.1002/9781119226079.ch12.

[38] G. Sabaliauskaite and A. Mathur, "Aligning Cyber-Physical System Safety and Security", *Complex System Design & Management*, Cham, Springer, 2021, pp. 41-53, doi: 10.1007/978-3-319-12544-2_4.

[39] J. Wang, T. Zhang, J. Song, N. Sebe and H. Shen, "A Survey on Learning to Hash", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 4, 2018, pp. 769-790, doi: 10.1109/tpami.2017.2699960.

[40] M. Ebrahim, S. Khan and U. B. Khalid, "Symmetric Algorithm Survey: A Comparative Analysis", *International Journal of Computer Applications*, Vol. 61, No. 20, 2013, pp. 12-19, doi: 10.5120/10195-4887.

[41] M. J. Moayed, A. A. A. Ghani and R. Mahmod, "A Survey on Cryptography Algorithms in Security of Voting System Approaches", in *2008 International Conference on Computational Sciences and its Applications*, Perugia, Italy, 2008, pp. 190-200, doi: 10.1109/ICCSA.2008.42.

[42] D. Lee, "Hash Function Vulnerability Index and Hash Chain Attacks", in *2007 3rd IEEE Workshop on Secure Network Protocols*, Beijing, China, 2007, pp. 1-6, doi: 10.1109/NPSEC.2007.4371616.

[43] F. Breitinger and H. Baier, "Properties of a Similarity Preserving Hash Function and Their Realization in SDhash", in *2012 Information Security for South Africa*, Johannesberg, South Africa, 2012, pp. 1-8, doi: 10.1109/ISSA.2012.6320445.

[44] S. El Moumni, M. Fettach and A. Tragha, "High Throughput Implementation of SHA3 Hash Algorithm on Field Programmable Gate Array (FPGA)", *Microelectronics Journal*, Vol. 93, 2019, pp. 1-8, doi: 10.1016/j.mejo.2019.104615.

[45] H. Choi and S. Seo, "Fast Implementation of SHA-3 in GPU Environment", *IEEE Access*, Vol. 9, pp. 144574-144586, 2021, doi: 10.1109/access.2021.3122466.

[46] S. Rahaman, N. Meng and D. Yao, "Tutorial: Principles and Practices of Secure Crypto Coding in Java", in *2018 IEEE Cybersecurity Development (SecDev)*, Cambridge, MA, 2018, pp. 122-123, doi: 10.1109/SecDev.2018.00024s.

[47] A. Baksi, S. Bhasin, J. Breier, D. Jap and D. Saha, "Fault Attacks in Symmetric Key Cryptosystems", *IACR Cryptology ePrint Archive*, 2020, pp. 1-24.

[48] P. K. Gundaram, A. Naidu Tentu and N. B. Muppalaneni, "Performance of Various SMT Solvers in Cryptanalysis", in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, 2021, pp. 298-303, doi: 10.1109/ICCCIS51004.2021.9397110.