Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 4, August 2021, pp. 2144~2151
ISSN: 2302-9285, DOI: 10.11591/eei.v10i4.2366

Improved MSHA-1 algorithm with mixing method

Rogel L. Quilala, Theda Flare G. Quilala
College of Computer Studies, Tarlac State University Tarlac City, Philippines

Article Info

Article history:

Received Mar 13, 2020
Revised Apr 29, 2021
Accepted May 24, 2021

Keywords:

Avalanche
Hashing
Message digest
SHA

ABSTRACT

Recently, a modified SHA-1 (MSHA-1) has been proposed and claimed to have better security performance over SHA-1. However, the study showed that MSHA-1 hashing time performance was slower. In this research, an improved version of MSHA-1 was analyzed using avalanche effect and hashing time as performance measure applying 160-bit output and the mixing method to improve the diffusion rate. The diffusion results showed the improvement in the avalanche effect of the improved MSHA-1 algorithm by 51.88%, which is higher than the 50% standard to be considered secured. MSHA-1 attained 50.53% avalanche effect while SHA1 achieved only 47.03% thereby showing that the improved MSHA-1 performed better security performance by having an improvement of 9.00% over the original SHA-1 and 3.00% over MSHA-1. The improvement was also tested using 500 random string for ten trials. The improved MSHA-1 has better hashing time performance as indicated by 31.03% improvement. Hash test program has been used to test the effectiveness of the algorithm by producing 1000 hashes from random input strings and showed zero (0) duplicate hashes.

Corresponding Author:

Rogel L. Quilala
College of Computer Studies
Tarlac State University
San Isidro extension campus, Tarlac City, Philippines
Email: rlquilala@tsu.edu.ph

1. INTRODUCTION

Cryptographic hash algorithms perform a major part in information security-it is a basis for a secure network which includes checking of data integrity [1], [4]. Hash algorithms are used in ensuring data integrity [5]. In hashing, a single bit of change from the input shall produce an entirely different hash output value [6]. A probability of 50% is the most ideal result in considering a change in the hash output, also known as the avalanche effect, but a higher value signifies higher efficiency of the technique [7]. The message digest or hash value produced by a hash function provides an assurance that the data during transmission will not be altered, because it is computationally infeasible for two distinct messages to have equal hash value [8], [9]. The hash value is the condensed representation of the message and acts as a digital fingerprint of the message, in which only one hash value should be associated with the message [8]. If the hash values differ, the integrity of the message is compromised. Hash is typically applied as integrity verification of password protection, security in protocols, file transmission, tamper detection, and digital signature [10], [11].

The algorithms under Secure Hash Standard Federal Information Processing Standards Publication (FIPS PUB 180-4) are as follows: SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, and SHA-512/256. Outputs of these hash algorithms are 160, 224, 256, 384, 512, 224 and 256 bits, respectively [12]. The National Institute of Standards and Technology published the cryptographic hash function Secure Hash Algorithm 1 (SHA-1) as (FIPS PUB 180-1) [13] and still belongs to (SHS) (FIPS PUB 180-4) [12]. The

SHA-1 algorithm is fast and has 160-bit output [14] and it is commonly used hash algorithm in different applications, and is as follows: PGP, SSL/TLS, Digital Signatures, and SSH [1], [15] because of its robustness and time efficiency [16]. In 2017, 21% of websites were recorded that SHA1 is still using for the signing of a website's certificate [17]. The SHA-1 with 160-bit hash value is widely used for file fingerprint and verification [18].

SHA-1 is considered simple, fast and the most widely used hash algorithm [11] because of robustness and time efficiency. However, it suffers from some unexpected weaknesses in the internal function and has a low avalanche effect [1], [19]-[21] thus leading to compromised data integrity. Thus, the modification of the hash function is essential in order to achieve higher avalanche effect diffusion [19], [22], [23]. Enhancements on SHA-1 were made to have better diffusion [24]-[25] but the simulation of results did not show the avalanche effect or have shown lower avalanche effect. The addition of MD5 hash to SHA-1 was also studied [22] but this approach has become suspect since MD5 has been completely broken [26]-[27]. Another study enhanced SHA-1 by modifying the message digest to increase the avalanche effect, but results revealed an avalanche effect that is lower than the ideal standard of 50% and has not included the actual messages used [28]. Another study has enhanced and modified the SHA-1 algorithm (MSHA-1). The study increased the hash value by adding 32-bit and modified the internal function in attaining better diffusion. Testing showed better diffusion, but hashing time of MSHA-1 was slower by 312.50% [29].

This study improves the MSHA-1 by modifying the output bits from 192-bit to 160-bit, with the purpose of improving the hashing time and keeping the mixing method in the compression function. The specific objectives were as follows: To determine the avalanche effect of the hashing algorithm with 160-bit output and mixing method; to evaluate the improvement of enhanced MSHA-1 over the MSHA-1 in terms of avalanche effect; to compare the performance of improved MSHA-1 against the MSHA-1 in terms of hashing time; to test the effectiveness of the improved MSHA-1 by using a hash test program. The main contribution of this work is beneficial to other researchers, because the improvement made on the MSHA1 by modifying the output 192-bit to 160-bit and the increase of avalanche effect of the improved MSHA-1, therefore when used as the hashing mechanism in their chosen application, will increase security and hashing time performance.

2. RESEARCH METHOD

2.1. Modification of MSHA-1 algorithm design

a. Modification on MSHA-1 was incorporated and described in detail below. The improved MSHA-1 160-bit hash value with the mixing method

The modification on MSHA-1. The working 32-bit variables A, B, C, D, E were used to produced 160-bit hash value, and the mixing method is used in every round to the compression function. This method will accept input from variables A, C, and D, and will have a new value to the output variables A', C', and D', as shown in Figure 1.

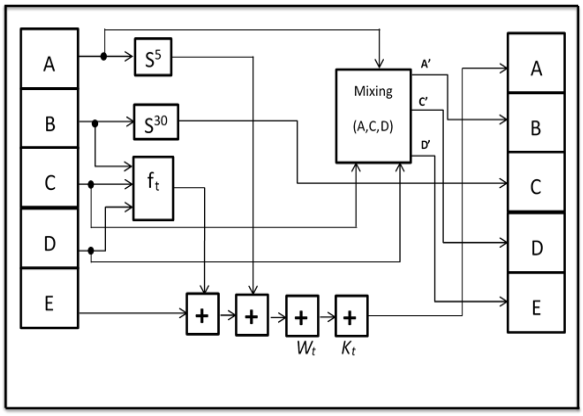

Figure 1. Improved MSHA-1 algorithm with the mixing method

b. The improved MSHA-1 algorithm process

– The padding of the message:

The length in multiple of 512 bits is the padded message. Number one (1) followed by zero (0) bit was added to the original message, and zero (0) number will depend on the message's original length. The binary representation of the original message's length is the last 64 bits of the previous 512-bit block.

– The constants and functions:

The Key_0, Key_1, Key_2 … Key_{79} and logical functions $funct_0$, $funct_1$ to $funct_{79}$ were used as constant words, and each $func_t$, $0 \leq t \leq 79$, the operation of three (3) 32-bit words and the output is 32-bit word. The functions and the constants used is shown as:

$func_t(B,C,D) = (B \&\& C) \| ((! B) \&\& D)$ Key_t = 5A827999	$0 \leq t \leq 19$
$func_t(B,C,D) = B \wedge C \wedge D$ Key_t = 6ED9EBA1	$20 \leq t \leq 39$
$func_t(B,C,D) = (B \&\& C) \| (B \&\& D) \| (C \&\& D)$ Key_t = 8F1BBCDC	$40 \leq t \leq 59$
$func_t(B,C,D) = B \wedge C \wedge D$ Key_t = CA62C1D6	$60 \leq t \leq 79$

– Computing the 160 Message Digest with mixing method:

a. The 16 words (Mi) were split by using these variables, W_0, ... W_{15}
b. When t = 16 to 79, do $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$.
c. Then let $A=H_1$, $B=H_2$, until $E=H_5$, counter = m
d. Set t=0 then do the following until t=79
   mix= mixing(A, C, D)
   A'=mix;C'=mix;D'=mix
   $T1 = S^5(A) + f_t(B, C, D) + E + W_t + K_t$;
   E = D'; D = C'; $C = S^{30}(B)$; B = A'; A =T1
e. nb+= m, then do $H_1 = (H_1 + A) \wedge nb$, $H_2 = (H_2 + B) \wedge nb$, $H_3 = (H_3 + C) \wedge nb$, $H_4 = (H_4 + D) \wedge nb$, $H_5 = (H_5 + E) \wedge nb$,
f. The algorithm will produce 160-bit hash value.

2.2. Testing

2.2.1. Avalanche effect with the use of mixing method

The MSHA-1 algorithm was modified using visual basic for applications (VBA). From here, the modifications were inserted. Different message sets were considered during the evaluation of improved MSHA-1. From these different message sets, the avalanche test was performed. Avalanche effect computation is shown in (1) and (2).

$$P = \left(\frac{\bar{B}}{l}\right) \times 100\% \quad (1)$$

$$\bar{B} = \frac{1}{N}\sum_{i=1}^{N} B_i \quad (2)$$

2.2.2. Improvement of modified SHA-1 over SHA-1 using different message sets

a. Message sets

For performance analysis, the study considered different message sets during the testing. Description of the various message sets was as follows:

Message set 1: message input of 1-bit change.

First message:

"@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA".

Second message:

"CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"[22].

Message set 2: Difference in only a few bits using the 64-character message set. Dissimilar characters were inserted at the 1st, 33rd, and 64th as shown in Table 1 [30].

Message set 3: Input 1:"abc123_owlstead_1255" and Input 2 "abc123_owlstead_59131" [31].

Message set 4: length differences as input [32], The message pattern was as follows: "a", "a a" and "a a a". The message pattern was shown in Table 2.

Message 5: Strings of a random message. The message was tested with the following input of characters 0...9, a...z, and A... Z from 500 random strings generated hash values [33].
Message set 6: Common substrings arranged in different orders, as shown in Table 3 [32].

Table 1. 64-character message set in difference of a few bits

No.	Message Input
1	@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
2	CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
3	EAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
4	IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
5	QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
6	aAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
7	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
8	ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
9	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
10	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
11	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
12	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
13	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
14	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
15	AAAAAAAAAAAAAAAAAAAAAAAAAAAAaAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
16	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
17	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@
18	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC
19	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE
20	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAI
21	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQ
22	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAa
23	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
24	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÁ

Table 1. The length different

No.	Message Input
1	a
2	a a
	.....
23	a a a a a a a a a a a a a a a a a a a a a a a
24	a a a a a a a a a a a a a a a a a a a a a a a a

Table 3. Common substrings arranged in different orders

No.	Message Input
1	The quick brown fox jump over the lazy dog
2	quick brown fox jump over the lazy dogThe
3	brown fox jump over the lazy dogThe quick
4	fox jump over the lazy dogThe quick brown
5	jump over the lazy dogThe quick brown fox
6	over the lazy dogThe quick brown fox jump
7	the lazy dogThe quick brown fox jump over
8	lazy dogThe quick brown fox jump over the
9	dogThe quick brown fox jump over the lazy
10	dog lazy the over jump fox brown quick The
11	The dog lazy the over jump fox brown quick
12	quick The dog lazy the over jump fox brown
13	brown quick The dog lazy the over jump fox
14	fox brown quick The dog lazy the over jump
15	jump fox brown quick The dog lazy the over
16	over jump fox brown quick The dog lazy the
17	the over jump fox brown quick The dog lazy
18	lazy the over jump fox brown quick The dog
19	The dog quick lazy brown the fox over jump
20	dog quick lazy brown the fox over jump The
21	quick lazy brown the fox over jump The dog
22	lazy brown the fox over jump The dog quick
23	brown the fox over jump The dog quick lazy
24	the fox over jump The dog quick lazy brown

b. Avalanche effect

The average avalanche effect of the improved MSHA-1 was compared to MSHA-1 and original SHA-1 using the different message sets. The experimental design was shown in Table 4. For each type of message, the avalanche percentage was computed for improved MSHA-1, MSHA-1, original SHA-1.

Table 4. Experimental design

	Message Type
1	1-bit change
2	The difference in a few bits
3	The difference in the last few bits
4	Length difference (24 Messages)
5	500 Random strings
6	Common substrings arranged in different orders

The percentage of improvement was also calculated to show a comparison in percentage. To calculate the percentage of change, the following formula is:

$$\%\ change = \frac{Original\ number - New\ number}{Original\ number} \times 100\% \quad (3)$$

2.2.3. Performance of improved MSHA-1 over MSHA-1 in terms of hashing time

Hashing time was also recorded using 500 random strings for ten trials (n=10). The average time was computed in milliseconds.

3. RESULTS AND DISCUSSION

3.1. Avalanche effect improved MSHA-1 160-bit hash value

The avalanche effect was computed using different message sets. The results were shown in Figure 2. The avalanche effect average of three algorithms. The results showed that the improved MSHA-1 was higher than the 50% ideal value by having 51.88%. The increase in the avalanched effect was due to the improvement and modification made.

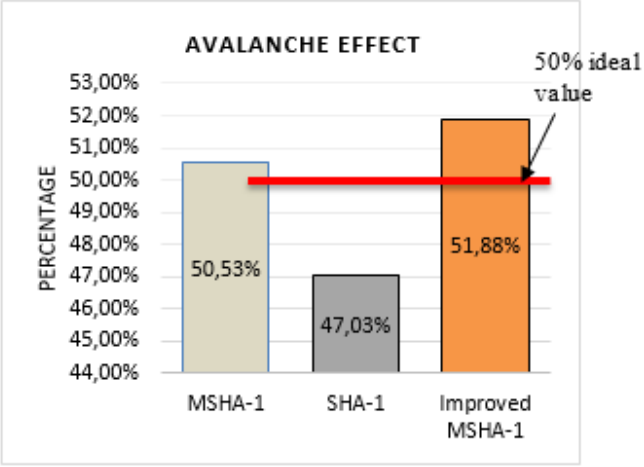

Figure 2. Average avalanche effect of improved MSHA1, MSHA-1 and SHA-1

3.2. Enhancement of improved MSHA-1 over MSHA-1 using message sets

3.2.1. Avalanche effect

Table 5 shows the avalanche percentage of the different message sets. For message type one, the improved MSHA-1 achieved 51.88%, while MSHA1 obtained 51.56%, and SHA-1 got 45.63%, as seen in Table 4. For message type two, the avalanche effect of the improve MSHA-1 acquired 50.49%, while MSHA-1 attained 50.09%, and SHA-1 got 48.37%. For message three, the improved MSHA-1 achieved

57.50%, which was very much higher than the 50% ideal value, while MSHA-1 got 50.00% and SHA-1 got only 38.75%, which was extremely lower than the ideal value. For message type four, the improved MSHA-1 achieved 50.49%, MSHA-1 attained 51.13%, and SHA-1 got 49.76%. For message type five, the improved MSHA-1 achieved 49.99%, and MSHA-1 got 50.19%, and SHA-1 attained 49.90%. Lastly, for message type 6, the improved MSHA-1 achieved 50.95%, while MSHA-1 attained 50.23%, and SHA-1 got 49.76%. Improved MSHA-1 has better security performance by having an avalanche effect improvement of 9% over SHA-1 and 3% over MSHA-1, as shown by the average avalanche value of 51.88% using the improved MSHA-1, 50.53% using the MSHA-1 and 47.03% using SHA-1.

Table 5. Avalanche effect of different message sets of MSHA-1, SHA-1 and improved MSHA-1

	Message Type	Avalanche effect (%)		
		MSHA-1	SHA-1	Improved MSHA-1
1	1-bit change	51.56%	45.63%	51.88%
2	The difference in only a few bits	50.09%	48.37%	50.49%
3	The difference in the last few bits	50.00%	38.75%	57.50%
4	Length difference (24 Messages)	51.13%	49.76%	50.49%
5	500 Random strings	50.19%	49.90%	49.99%
6	Common substrings arranged in different orders	50.23%	49.76%	50.95%
	Average (%)	50.53%	47.03%	51.88%

The hash function test program was used to test the effectiveness of the improved MSHA-1, as shown in Figure 3 [34]. The function will count the duplicate hash values of the one thousand inputted hashes that have been produced by the improved MHSA-1. Based on the output, it was clearly shown that there were no duplicates hash values found from the hash values produced by the improved MSHA-1 algorithm.

```
Hash[995]: 63019feb7182de69f0f5ece3716374cf8b31266c
Hash[996]: aeeeba5ecbb35391136dbbfc46913646b11df2f8
Hash[997]: 0b87ee65db1f4174ad1ad3e9127c9bdef0acf60a
Hash[998]: 7e888ecda2bd08f4472d646a116260dbcfd29dbf
Hash[999]: 4521e04e739e52815becb66ecdbbd1c529f86705
Total number of hash values: 1000
Duplicate hash values: 0
done
```

Figure 3. Improved MSHA-1 hash function test

3.3. Performance of MSHA-1 vs SHA-1 in terms of hashing time

Hashing time was also recorded using 500 random strings for ten trials (n=10). The average time was computed in milliseconds. Figure 4 shows the average hashing time of the improved MSHA-1 and MSHA-1. The time difference of the improved MSHA-1 and MSHA-1 was computed using 500 random strings as a message set. Time was noted for ten trials, as shown in Table 6. The average time noted for improved MSHA-1 was 878.85ms, while MSHA-1 was 1151.52ms. The hashing time of improved MSHA-1 has better hashing time performance, as indicated by a 31.03% improvement.

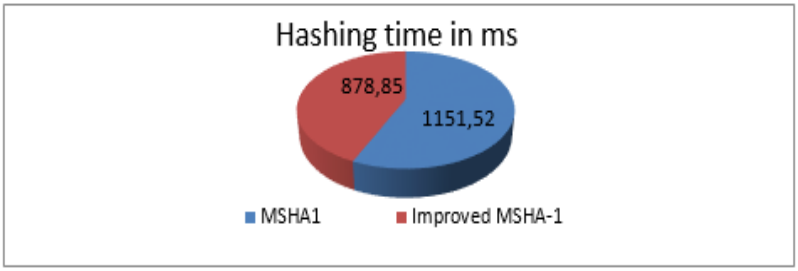

Figure 4. Average hashing time of the 24 messages in milliseconds

Table 6. Average hashing time of the 500 random strings in milliseconds

Trials n=10	MSHA-1	Improved MSHA-1
1	1484.38	765.63
2	1109.38	953.13
3	1007.81	968.75
4	1015.63	867.19
5	1078.13	843.75
6	1023.44	882.81
7	1085.94	828.13
8	1031.25	875.00
9	1156.25	828.13
10	1523.00	976.00
Average (ms)	1151.52	878.85

4. CONCLUSION

From the summary of findings and results presented in the previous section, the following can be concluded that improved MSHA-1 with 160-bit hash value and using the mixing method resulted in an avalanche effect of 51.88%. Compared to the original SHA-1 and MSHA-1, the improved MSHA-1 performed better by having an avalanche effect improvement of 9.00 % over the SHA-1 and 3% over MSHA-1. Compared to the MSHA-1, the hashing time of improved MSHA-1 has a better performance by having a 31.03% improvement. The improved MSHA-1 showed no duplicates using the hash test program. These results show that the modification and improvement made on the MSHA-1 algorithm enhanced security and improved the hashing time performance. As an offshoot of this study, further researches may be undertaken by modifying MSHA-1 hash through lessening the number of rounds to improve more the hashing time performance.

REFERENCES

[1] N. Kishore and B. Kapoor, "Attacks on and advances in secure hash algorithms," *IAENG Int. J. Comput. Sci.*, vol. 43, no. 3, pp. 326-335, 2016.

[2] M. A. Alahmad, "Design of a New Cryptographic Hash Function - Titanium," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 10, no. 2, pp. 827-832, 2018, doi: 10.11591/ijeecs.v10.i2.pp827-832.

[3] K. M. Lee, K. M. Lee, and S. H. Lee, "Remote data integrity check for remotely acquired and stored stream data," *J. Supercomput.*, vol. 74, no. 3, pp. 1182-1201, 2018, doi: 10.1007/s11227-017-2117-4.

[4] D. Wang, Y. Jiang, H. Song, F. He, M. Gu, and J. Sun, "Verification of Implementations of Cryptographic Hash Functions," *IEEE Access*, vol. 5, pp. 7816-7825, 2017, doi: 10.1109/ACCESS.2017.2697918.

[5] I. Alsmadi and M. Zarour, "Online integrity and authentication checking for Quran electronic versions," *Appl. Comput. Informatics*, vol. 13, no. 1, pp. 38-46, 2017, doi: 10.1016/j.aci.2015.08.001.

[6] H. Liu, A. Kadir, and J. Liu, "Keyed Hash Function Using Hyper Chaotic System with Time-Varying Parameters Perturbation," *IEEE Access*, vol. 7, no. c, pp. 37211-37219, 2019, doi: 10.1109/ACCESS.2019.2896661.

[7] G. K. Sodhi and G. S. Gaba, "An efficient hash algorithm to preserve data integrity," *J. Eng. Sci. Technol.*, vol. 13, no. 3, pp. 778-789, 2018.

[8] R. P. Arya, U. Mishra, and A. Bansa, "A Survey on Recent Cryptographic Hash Function Designs," *Int. j. emerg. trends technol. comput. sci.*, vol. 2, no. 1, pp. 1-6, 2013, doi: 10.1.1.587.7541.

[9] X. Wang, S. Wang, N. Wei, and Y. Zhang, "A Novel Chaotic Image Encryption Scheme Based on Hash Function and Cyclic Shift," *IETE Tech. Rev. (Institution Electron. Telecommun. Eng. India)*, vol. 36, no. 1, pp. 39-48, 2019, doi: 10.1080/02564602.2017.1393352.

[10] W. Chankasame and W. San-Um, "A chaos-based keyed hash function for secure protocol and messege authentication in mobile ad hoc wireless networks," in *2015 Science and Information Conference (SAI)*, Jul. 2015, pp. 1357-1364, doi: 10.1109/SAI.2015.7237319.

[11] R. K. Ibraheem, R. A. J. Kadhim, and A. S. H. Alkhalid, "Anti-collision enhancement of a SHA-1 digest using AES encryption by LABVIEW," *2015 World Congr. Inf. Technol. Comput. Appl.*, pp. 1-6, 2015, doi: 10.1109/WCITCA.2015.7367044.

[12] Q. H. Dang, "Secure Hash Standard," Gaithersburg, MD, Jul. 2015. doi: 10.6028/NIST.FIPS.180-4.

[13] NIST, "FIPS 180-1 - Secure Hash Standard," *FIPS PUB 180-1*, no. April 17, 1995.

[14] P. Garg and N. Tiwari, "Performance Analysis of SHA Algorithms (SHA-1 and SHA-192): A Review," *Int. j. comput. technol. electron. eng.*, vol. 2, no. 3, pp. 130-132, 2012.

[15] T. Mantoro and A. Zakariya, "Securing E-mail Communication Using Hybrid Cryptosystem on Android-based Mobile Devices," *TELKOMNIKA (Telecommunication, Comput. Electron. Control.)*, vol. 10, no. 4, pp. 827-834, 2012, doi: 10.12928/telkomnika.v10i4.397.

[16] K. Saravanan and A. Senthilkumar, "Theoretical Survey on Secure Hash Functions and issues," *International Journal of Engineering Research & Technology*, vol. 2, no. 10, pp. 1150-1153, 2013.

[17] Venafi, "Venafi Research: Twenty-One Percent of Websites Are Still Using Insecure SHA-1 Certificates and Putting Users at Risk," *Venafi Press Release*, 2017. https://www.venafi.com/news-center/press-release/venafi-

research-twenty-one-percent-of-websites-are-still-using-insecure.
[18] M. Stevens and D. Shumow, "Speeding up detection of SHA-1 collision attacks using unavoidable attack conditions," *USENIX Secur.*, vol. 2017, p. 173, 2017, [Online]. Available: http://dblp.uni-trier.de/db/journals/iacr/iacr2017.html#StevensS17.
[19] A. Kumarkasgar, J. Agrawal, and S. Shahu, "New modified 256-bit MD5 Algorithm with SHA Compression Function," *Int. J. Comput. Appl.*, vol. 42, no. 12, pp. 47-51, Mar. 2012, doi: 10.5120/5748-7956.
[20] P. Karpman, T. Peyrin, and M. Stevens, "Practical Free-Start Collision Attacks on 76-step SHA-1," *Annual Cryptology Conference*, vol. 2012, 2015.
[21] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," *Adv. Cryptol. - CRYPTO 2005*, no. 90304009, pp. 17-36, 2005, doi: 10.1007/11535218_2.
[22] G. Gupta and S. Sharma, "Enhanced SHA-192 Algorithm with Larger Bit Difference," *2013 International Conference on Communication Systems and Network Technologies*, 2013, pp. 152-156, doi: 10.1109/CSNT.2013.42.
[23] X. Xu, Q. Zhao, and C. Li, "Advanced Framework for Iterative Hash Functions," *2012 Int. Conf. Comput. Sci. Electron. Eng.*, pp. 599-602, 2012, doi: 10.1109/ICCSEE.2012.140.
[24] C. C. G. San Jose, B. T. Tanguilig III, and B. D. Gerardo, "Enhanced SHA-1 on Parsing Method and Message Digest Formula," *Proceedings of the Second International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA2015)*, Manila, pp. 1-9, 2015.
[25] L. Thulasmani and M. Madheswaran, "Security and Robustness Enhancement of Existing Hash Algorithm," *2009 Int. Conf. Signal Process. Syst.*, pp. 253-257, 2009, doi: 10.1109/ICSPS.2009.49.
[26] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.," *IACR Cryptol. ePrint Arch.*, vol. 5, no. October, pp. 5-8, 2004, [Online]. Available: http://web.mit.edu/fustflum/documents/crypto.pdf.
[27] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," *Adv. Cryptol. - EUROCRYPT 2005*, pp. 19-35, 2005, doi: 10.1007/11426639_2.
[28] S. Verma and G. S. Prajapati, "Robustness and security enhancement of SHA with modified message digest and larger bit difference," *2016 Symp. Colossal Data Anal. Netw.*, pp. 1-5, 2016, doi: 10.1109/CDAN.2016.7570959.
[29] R. L. Quilala, A. M. Sison, and R. P. Medina, "Modified SHA-1 Algorithm," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 11, no. 3, pp. 1027-1034, 2018, doi: 10.11591/ijeecs.v11.i3.pp1027-1034.
[30] H. M. Elkamchouchi, M. E. Nasr, and R. I. Abdelfatah, "A new secure and fast hashing algorithm (SFHA-256)," in *2008 National Radio Science Conference*, Mar. 2008, no. Nrsc, pp. 1-8, doi: 10.1109/NRSC.2008.4542348.
[31] M. Bodewes, "SHA-1 Collision for First 32-bits for Two Different Message," 2016, [Online]. Available: https://crypto.stackexchange.com/questions/34575/sha1-collision-for-first-32-bits-for-two-different-messages.
[32] C. Estébanez, Y. Saez, G. Recio, and P. Isasi, "Performance of the most common non-cryptographic hash functions," *Softw. Pract. Exp.*, vol. 44, no. 6, pp. 681-698, Jun. 2014, doi: 10.1002/spe.2179.
[33] "Text Mechanic - Random String Generator Tool." [Online]. Available: https://textmechanic.com/text-tools/randomization-tools/random-string-generator/
[34] N. Mudge, "Hash Function Test Program,". [Online]. Available: https://gist.github.com/david7482/80c8c3c8962c1621c072

BIOGRAPHIES OF AUTHORS

Dr. Rogel L. Quilala is an assistant Professor of ICT at the Tarlac State University-College of Computer Studies. He received his Doctor in Information Technology (DIT) degree from Technological Institute of the Philippines (TIP). He has published several research papers in Scopus-indexed journals. He has also presented several papers in international conferences and has been awarded Best in Presentation at the 2nd International Conference on Imaging, Signal Processing and Communication (ICISPC 2018) in Kuala Lumpur, Malaysia.

Dr. Theda Flare G. Quilala is currently an Associate Professor in the College of Computer Studies at Tarlac State University, Tarlac City, Philippines. A Doctor of Information Technology graduate at Technological Institute of the Philippines and a CHED K-12 scholar. Research interest includes security, data mining, and algorithms.