Indonesian Journal of Electrical Engineering and Computer Science
Vol. 11, No. 3, September 2018, pp. 1027~1034
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v11.i3.pp1027-1034

Modified SHA-1 Algorithm

Rogel L. Quilala[1], Ariel M. Sison[2], Ruji P. Medina[3]
[1,3]Technological Institute of the Philippines, 938 Aurora Blvd., Cubao, Quezon City, Philippines
[2]Emilio Aguinaldo College, 1113-1117 San Marcelino St., Paco, Manila 1000, Philippines

Article Info

Article history:

Received Feb 27, 2018
Revised Apr 21, 2018
Accepted Jun 14, 2018

Keywords:

Hash
Data integrity
Security
Cryptography
Avalanche

ABSTRACT

Hashes are used to check the integrity of data. This paper modified SHA-1 by incorporating mixing method in every round for better diffusion. The modification increased the hash output to 192-bits. Increasing the output increases the strength because breaking the hash takes longer. Based on the different message types, avalanche percentage of modified SHA-1 showed better diffusion at 51.64%, higher than the target 50%, while SHA-1 achieved 46.61%. The average execution time noted for modified SHA-1 is 0.33 seconds while SHA-1 is 0.08 seconds. Time increases as the number of messages hashed increases; the difference is negligible in fewer messages. On character hits, that is - no same character in the same position, modified SHA-1 achieved lower hit rate because of the mixing method added. The modifications' effectiveness was also evaluated using a hash test program. After inputting 1000 hashes from random strings, the result shows no duplicate hash.

Corresponding Author:

Rogel L. Quilala,
Technological Institute of the Philippines
938 Aurora Blvd., Cubao, Quezon City, Philippines.
Email: rlquilala@gmail.com

1. INTRODUCTION

In checking data integrity, cryptographic hash algorithms performs significant part to information security [1], [2]. Data files used hashes for verifying its integrity, where a little change will cause a different hash value [3].Hash assure that the recipient obtained the message sent by the source and that there is no form of alteration done during transmission [4]. The representation of the message in compressed form is called message digest or hash value. Hash value act as a digital fingerprint of the message or file, wherein a message can only have one distinct hash value thus no two messages should have the same hash [4]. If the hash value differs, hackers did alterations during transit resulting in the compromised integrity of the message. Electronically transmitted files, digital signature, tamper detection, password protection, and security in protocols apply hash for integrity verification [5], [6].

Seven approved hash algorithms are in Secure Hash Standard (SHS) Federal Information Processing Standards Publication (FIPS PUB 180-4) namely: SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, and SHA-512/256 with hash length of 160, 224, 256, 384, 512, 224 and 256 bits, respectively [7]. SHA family uses the traditional iterative structure by Merkle-Damgard (M-D) [8], [9]. Even though M-D construction ensures the security of hash functions, it suffers from some vulnerabilities due to structural weakness [10]. That is why more hash functions that address shortcomings in the M-D construction are being suggested incorporating minimal changes [11] such as wide and double pipe construction, 3C, prefix, chop, sponge, and others each exhibiting their strengths and weaknesses. In this paper, the construction will be modified by adding a counter and XORing the number to the intermediate hash value. With this additional process, the modified SHA-1 strengthened the construction because of the addition of the counter which changes at every step.

National Institute of Standards and Technology (NIST) published Secure Hash Algorithm 1 (SHA-1) as a cryptographic hash function [7], [12]. SHA-1 produces 160-bit hash value and is considered fast [13]. It is the most widely used hash algorithm in a vast range of applications such as Digital Signatures, TLS/SSL, SSH and PGP [14]-[16] due to its time efficiency and robustness [17]. At present, 21% of websites in the world still use SHA-1 in signing certificates [18]. SHA-1 based fingerprint is used widely and supported for verification [19].

Other hash functions also exist such as MD5 by Ronald L. Rivest released in 1992 that can compress any data length to a hash value of 128bits [20], but real collision broke MD5 entirely in 2004 [21], [22]. SHA-0 in 1993 is an MD4 hash function used for authentication, is believed to be not safe after several successful collision attacks in 2004 and 2005 [1]. SHA-2 and SHA-3 provide more extended hash value that is more complicated to break [11], [13], but they are more complex and not as time efficient as SHA-1 [14] [23], [24]. The increased number of rounds in SHA3 makes it less susceptible to collision resistance and preimage resistance attacks when measured against SHA2, MD5, and SHA1 and others [25] but the use of a sponge function construction can be considered neither as an advantage nor a disadvantage because this function is a new construction that is not yet very well analyzed [26].

Though SHA-1 is popular, widely used and accepted as standard by NIST. Some noted that it does not seem to offer sufficient avalanche effect with regards to the distribution of the input differences, while other noted some unexpected weaknesses in the construction of all the step updating functions [1], [27]. This problem will lead to the possibility of having two different input that will yield the same output value in the middle of algorithm or compression function [20] [28]. Therefore, it is necessary to design a function with better diffusion to spread the output in each round and prevent the same output in the next coming stages [20], [29]-[30].

Several studies made several enhancements on SHA-1 to attain additional diffusion [31], [32] but did not show the bit-difference on the simulation of result or have shown lower bit difference. One study has added the MD5 hash to SHA-1 [29] that indicates that the bit-difference of SHA-192 is lower than SHA-160. This approach might suffer from the same weakness as that of MD5 [21], [22]. Others have not included the actual message in the comparison of bit-difference. [23]. Therefore, the researcher has decided to improve SHA-1 algorithm by increasing hash size output from 160 to 192 bits and provide better diffusion. Another enhancement of SHA-1 makes use of 320-bit hash by doubling the message digest size and hash size [14]. This enhancement decreases the chances of the collision, but this approach requires more processing time since it makes use of a higher block size. Notice that all enhancements made on SHA 1 uses the chaining variables A, C, and D in each round as is and is just shifted to the next chaining variables and sends it to the next round. From here, the researcher proposed to devise the mixing method to diffuse variables A, C, and D better for each iteration.

This study intended to modify SHA-1 algorithm by increasing the output to 192-bits and strengthening the hash function by adjusting the compression function through the incorporation of additional mixing method in every round with the intention of attaining better diffusion. The objectives of this study are to evaluate the performance of the modified SHA-1 through avalanche effect and to test the modified SHA-1 algorithm regarding time and message complexity.

The main impact of this work is the improvement of SHA-1 by introducing additional mixing method in every round to achieve better diffusion characteristics. The study will contribute to the improvement of the compression function used by SHA1 by increasing the output of the hash value to 192-bits to strengthen the algorithm. Higher time will be needed to break the hash.

2. RESEARCH METHOD

2.1. Research Procedure

Figure 1 shows the proposed modified SHA-1 construction with the counter. An added counter was XORed to the intermediate hash value. The addition of this process strengthened the M-D construction because of a number assigned to the counter that changes in every step. The counter will start at an initial value of zero and is incremented by 1 for every message block until the last block.

The proposed SHA algorithm of the compression function retained the eighty rounds. The modified SHA-1 increased the message digest from 160-bits to 192-bits to strengthen the algorithm. To achieve this, one additional chaining variables F was added. Next, F was XORed to the output of E before going to A. All researchers have used variables A, C, and D as is. In every round, these variables were injected into the mixing function to achieve better diffusion. The variables are mixed every round and send it to the next round. This mixing function guarantees that the input values will spread out thus promoting good diffusion in each round because the contents of the variables will not be the same in the coming rounds. Variable E goes to variable F after own addition operations. Figure 2 shows the proposed modification on SHA-1 with the added mixing method. In the proposed hash algorithm, we note significant changes in the elementary function.

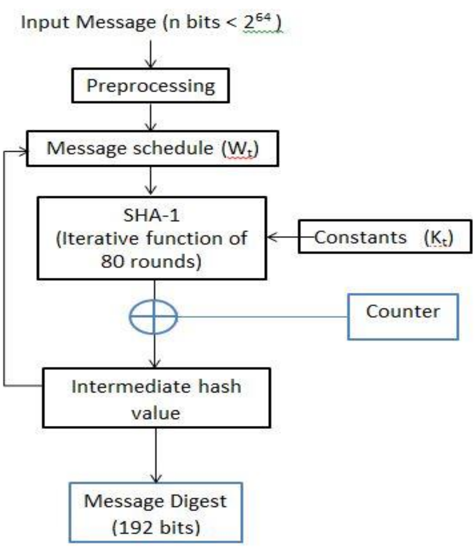

Figure 1. Proposed modification on SHA-1 construction

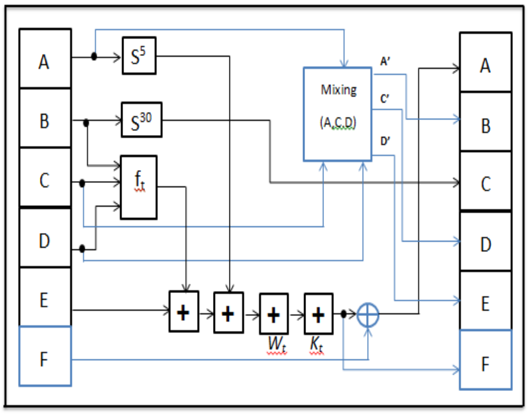

Figure 2. Proposed modification on SHA-1 compression with added mixing method

The modified SHA-1 follows the same step in SHA except for the computation of the message digest. The padded message is used to compute for the message digest. The computation uses two buffers (A, B, C, D, E, F and H_0, H_1, H_2, H_3, H_4, H_5). The first buffer uses five 32-bit words, and the second buffer comprises of eighty 32-bit words (W0, W1 ... W79). This process also uses TEMP1 and TEMP2 buffers. {Hj} are initialized before processing any blocks with values of 67452301, EFCDAB89, 98BADCFE, 10325476, C3D2E1F0, 40385172 (H_1-H_5). Let hash value length be m.

Modified SHA-1 steps to process the message in 16-word blocks:

a) Split Mi into 16 words starting from left to right, W_0, ... W_{15}
b) When t = 16 to 79, we 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_0, B=H_1, until F=H_5, counter = m
d) When t = 0 upto 79 do
mixedACD= mixingACD(A, C, D)

A’=mixedACD;C’=mixedACD;D’=mixedACd
TEMP1 = $S^5(A) + f_t (B, C, D) + E + W_t + K_t$;
TEMP2 = F xor TEMP1
E = D’; D = C’; C = $S^{30}(B)$; B = A’; A = TEMP2; F=TEMP1

e) counter+= m, then do $H_0 = (H_0 + A)$ xor counter, $H_1 = (H_1 + B)$ xor counter, $H_2 = (H_2 + C)$ xor counter, $H_3 = (H_3 + D)$ xor counter, $H_4 = (H_4 + E)$ xor counter, $H_5 = (H_5 + F)$ xor counter.

After processing M_n, these words represent the computed 192-bit hash value:

$H_0\ H_1\ H_2\ H_3\ H_4\ H_5$

The purpose of the Mixing (A, C, D) function is to accept the working variables A, C, and D as the input column then spread the bits out to different places in the output column A', C,' and D'. The mix is arranged from right to left in row-wise fashion as illustrated in Figure 3.

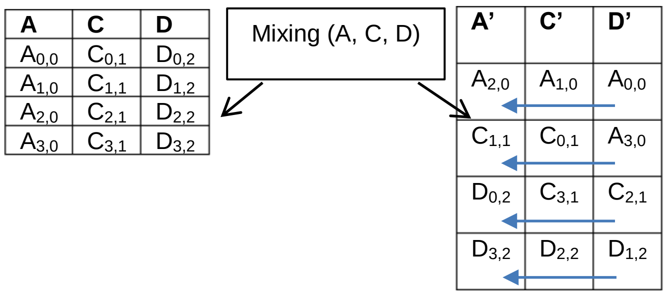

Figure 3. Mixing function

2.2 Evaluation Metrics

The performance of the modified SHA-1 was evaluated through avalanche effect, time and message complexity.

Avalanche effect is a suitable characteristic in a hash function which indicates that a change in the input bit of the hash results to a difference on the probability of the output bit. If the chance is close to 50%, the hash function is considered good. A 50% avalanche percentage shows that the difference of the output hash value and the input change is at least half and a probability higher than 50% displays improved statistical performance [33].

Time notes the speed to generate the hash in seconds. Classification of the message type is two message with 1-bit change, 24 messages with a difference in a few bits, two messages with distinction in the last few bits, length difference, and random strings. Performance of the hash function is also measured by comparing hash values with each other and then counting characters located at the same location with the same content [34], in this study referred to as character hit.

3. RESULTS AND ANALYSIS

For performance analysis, we consider different messages during the testing and time, and avalanche effect was noted for each test. The first message type is a 1-bit change in the message input. Consider the two message: Message 1: “The quick brown fox jumps over the lazy dog” and Message 2: “The quick brown fox jumps over the lazy mog”.

The second message type was tested using an input with a difference in only a few bits. Table 1 lists the twenty-four messages used. The researcher inserts different characters at the beginning, middle, and last.

For the third message, consider the two words: “abc123_owlstead_1255” and “abc123_owlstead_59131”.

The fourth message input is the length differences, that is the message “a a a” has a length of 5 versus message ”a a” which has a length of 3. The length of message considered was listed in Table 2

Table 1. Message inputs with a difference of a few bits

No.	Message Input
1	@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
2	CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
3	EAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
4	IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
5	QAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
6	aAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
7	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
8	ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
9	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
10	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
11	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
12	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
13	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
14	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
15	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAaAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
16	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
17	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA@
18	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC
19	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE
20	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAI
21	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQ
22	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAa
23	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
24	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAÁ

Table 2. Message inputs with different length

No.	Message Input
1	a
2	a a
3	a 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

The fourth type is a random string of message. For this test, the message consists of characters a...z, A...Z, and 0...9. An online tool helps generate hashes from 500 random strings each of length 64 [35].

Table 3. Summary of results

	Message Type	Avalanche (%)		Time (seconds)	
		MSHA-1	SHA-1	MSHA-1	SHA-1
1	Two messages with 1-bit change	56.77	46.25	0.02	0.02
2	24 messages w/diff. in a few bits	50.09	48.37	0.09	0.05
3	Two messages w/diff. in last few bits	50.00	38.75	0.02	0.02
4	Length difference	51.13	49.76	0.06	0.05
5	Random strings	50.19	49.90	1.28	0.28
	Average (%)	51.64	46.61	0.33	0.08

For message type 1, the proposed modified SHA-1 achieved 56.77% while SHA-1 obtained 46.25%. Hashing time for both tests is 0.02 seconds. For message type 2, avalanche effect of the proposed modification on SHA-1 obtained 50.09%. The original SHA-1 attained 48.37%, slightly lower than the desired 50%. Concerning the execution time, as reflected in Table 3, it took the modified SHA-1 0.09 seconds to complete while SHA-1 took 0.05 seconds. The modified SHA-1 is a bit higher. For message type 3, the modified SHA-1 achieved exactly 50.00% while SHA-1 got 38.75%, which is significantly lower. Hashing time for both tests is 0.02 seconds. For message type 4, the modified SHA-1 achieved 51.13% while SHA-1 attained 49.76%. The hashing time shows 0.06 and 0.05 for modified SHA and SHA-1. There is a very minimal difference when it comes to the hashing time. Lastly, for message type 5, the modified SHA achieved 50.19% while SHA-1 arrived at 49.90%5. The modified SHA-1 hashed the random string of message in 1.28 seconds while SHA-1 produced the hash in 0.28. A difference of 1.00 seconds can be noted.

Based on the average, the avalanche effect of all has increased due to the modifications made. The testing showed better diffusion result because out of the five different message types, the average avalanche percentage of modified SHA-1 was 51.64% which is higher than the target 50% while SHA achieved only 46.61%. Regarding the time it takes to produce the hash, the time recorded was the same for two-message comparisons. An increase in time appears as the number of the message to be hashed enlarges. The average time noted for modified SHA-1 is 0.33 while SHA-1 is 0.08. The increment is mostly due to the added mixing method and XOR operation. Although the time associated with hashing a message using modified SHA is a bit higher, there is evident character hits as shown in Table 4.

Table 4. Summary of character hits

	Message Type	Total Character Hits		Max No. Of Equal Character Hits	
		MSHA-1	SHA-1	MSHA-1	SHA-1
1	1-bit change	0	0	0	0
2	24 messages w/diff. in a few bits	0	2	0	1
3	Two messages w/diff. in last few bits	0	5	0	5
4	Length difference	1	6	1	2
5	Random strings	38	48	1	2
	Average (%)	(2:5) 40	(4:5) 80		

In the modified SHA-1, message types 1, 2, and three doesn't have any character hits. For message type 4, out of the 24 hashes generated, there was one instance where the same character was at the same position. For message type 5, out of the 500 random messages, 38 hash pairs contains one character hit.

Character hits are noted more frequently in SHA-1. For message type 2, there were two hits recorded. There were five hits observed for message type 3 and the number of characters per hit ranges from 1-5 characters per hash. For message type 4, 6-character hits and the number of characters that match ranges from 1-2 per hash. For message 5, there were 44 hash pairs containing one character hit per hash and two hash pairs with 2 character hits for a total of 48 hits. Notice that the hits for the original SHA-1 are higher compared to the adjusted version.

The modified SHA-1 simulation indicates that out of the five message types, there were two instances where a character hit was noted (2:5 or 40%) while in SHA-1, character hits occur 4 out of the five different message types (4:5 or 80%). When considering the number of hits, the modified SHA-1 has a much lower hit rate compared to the original SHA-1 on all tests made and on all test cases.

1	934f46412dbdea31a1d91aa27b19835014ac7c55
2	a66aaff13b6012bf7bcdd8815e8d41f25ca72168
3	33660742c1e6ff3a86489535474613c2fd0935b8
4	2458cd342ab44954a261a44f94507dc1412262ac
5	7db41c9d2d390684b428680d1475b4e2ab854e08
6	f082a212742687a630ff4677c46f15f627490098
7	4547c3b7c7d43b64debe62b01767284901046aa2
8	f2d40a511fa54b686b7e0162529be0981957**12**25
9	30b86e44e6001403827a62c58b08893e77cf**12**1f
[illegible]	[illegible]
23	a320100894f91165b624376e4c8d01cea46374**24**
24	d091c57996091b918aaa76233bac434609734d**24**

Figure 4. Hash list for a message with the difference in a few bits

To understand character hits, using the 24 message inputs and their hash value in SHA-1, the researcher count the values that have the same hexadecimal value at the same position. Two hexadecimal value is equal to 1 hit. Using traditional SHA-1 as shown in Figure 4, the hash of the message having a difference in a few bits found two hits (Hash 8 and 9, hash 23 and 24). Figure 5 illustrates another example using two messages with the difference in a few bits. In the modified SHA-1, the computed hash found no values on the same location. In SHA-1, there are nine hexadecimal values or 5 ASCII characters located at the same place.

SHA-1 Hash value
992156e04e6511eb30830a8dbd9dfac852e38117
992156e0429eb1ebef46681aff430dc30cf24db5

Modified SHA-1 with mixing method hash value
871c0f73015f30ab2e460896621cdde1f71d43df0f7db022
a7ef3e8130bdbfc436ea902b9808b5a5e92688f60106270c

Figure 5. Hash list of two messages with the difference in a few bits

The hash value produced by the modified hash was also tested using a hash function testing program [36]. This program takes hash values and counts how many duplicates the hash function produces. 1000 hashes from random strings were generated using the modified SHA-1 algorithm, and after running the hash test, modified SHA-1 found no duplicates. Figure 6 shows the screenshot of the hash function test.

```
Hash[985]: ebe567b33a7c3752f30afead67b6d5398314c5c416b4d6b7
Hash[986]: 8f300d9b432fcadee469085f28c333c537ed2dde0b35715c
Hash[987]: ef44ad6acff8c554b6b117490d9485f5f2e3a3aa9f67f4a4
Hash[988]: 884329c658bf02a85dc58ebc4d87b21e71db0ce73b77c677
Hash[989]: 2c42d77536007c5a9e3cf12a4b8b8f427e58364db6057109
Hash[990]: 1fa8f2d493fd761bb5c2173c44d9c1b5febe6162ca592ca3
Hash[991]: 4f36cc7a38dffb028a9b90278e7be329d364551f62a8de35
Hash[992]: c066f410cf3fc5e30bbc8d8cf65677827e37204d4a29f700
Hash[993]: 8e0e883d50382dae79587e107eab9b6b9ff7dcc3c950637c
Hash[994]: d88dff37e0158c3ff1d0c3a536625e13f85c4993aaa4ab17
Hash[995]: 3cb2ed220498d01d2be8d5acc91274eafc6ef8ed8ef7fc38
Hash[996]: d660775c5475e9bc489d16ad7b1a53896c66e3c4cbc4379a
Hash[997]: dde7a18680c80dee66206ac01bd26516775bc097014fbb1
Hash[998]: c0b83e2300f9e5524880cf56fb19ac6a1a1749e5eb1022b0
Hash[999]: 503d2a025a97c955476784faef19265fdf78907797e6f7ee
Total number of hash: 1000
Duplicate hash: 0
done
```

Figure 6. Hash function test

4. CONCLUSION

This study intended to modify SHA-1 algorithm by increasing the output to 192-bits and strengthening the hash function by adjusting the compression function through the incorporation of additional mixing method in every round with the intention of attaining better diffusion. Looking at the results of the tests done, the modified SHA-1 have better diffusion compared to the original SHA-1. The diffusion is evident by the increase in the avalanche percentage. There is an increase in the avalanche percentage although the time also increased when messages increased. The additional mixing method and XOR operation contribute to the increment in time. It is also evident that the number of hits using the modified SHA-1 was minimal or lower compared to the original SHA-1 leading to no collision. Upon using the hash function testing program, the hash values found have no duplicates. Based on the results, the modified SHA-1 can be used to test the integrity of messages. Further improvement is suggested to minimize the time consumed by the modified SHA-1 hash by studying the effect of lessening the number of rounds.

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.
[3] 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.
[4] 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. 117–122, 2013.

[5] W. Chankasame and W. San-Um, “A chaos-based keyed hash function for secure protocol and messege authentication in mobile ad hoc wireless networks,” *Proc. 2015 Sci. Inf. Conf. SAI 2015*, pp. 1357–1364, 2015.
[6] 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.
[7] Q. H. Dang, “Secure Hash Standard,” Gaithersburg, MD, Jul. 2015.
[8] R. C. Merkle, “One Way Hash Functions and DES,” in *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, 1990, pp. 428–446.
[9] I. B. Damgård, “A Design Principle for Hash Functions,” *CRYPTO`89 Proceedings*, vol. 435. pp. 416–424, 1990.
[10] H. Tiwari and K. Asawa, “Building a 256-bit hash function on a stronger MD variant,” *Open Comput. Sci.*, vol. 4, no. 2, pp. 67–85, Jan. 2014.
[11] R. Sobti and G. Geetha, “Cryptographic hash functions: a review,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 461–479, 2012.
[12] NIST, “FIPS 180-1 - Secure Hash Standard,” *FIPS PUB 180-1*, no. April 17, 1995.
[13] P. Garg and N. Tiwari, “Performance Analysis of SHA Algorithms (SHA-1 and SHA-192): A Review,” *Int. J.*, vol. 2, no. 3, pp. 130–132, 2012.
[14] S. Rao, “Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity,” *Int. J. Comput. Appl.*, vol. 130, no. 8, pp. 25–27, 2015.
[15] R. A. N. Karthik, A.K. Parvathy, “Non-convex Economic Load Dispatch using Cuckoo Search Algorith,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 5, no. 1, pp. 48–57, 2017.
[16] 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.
[17] K. Saravanan and A. Senthilkumar, “Theoretical Survey on Secure Hash Functions and issues,” *Int. J. Eng. Res. Technol.*, vol. 2, no. 10, pp. 1150–1153, 2013.
[18] Venafi, “Venafi Research: Twenty-One Percent of Websites Are Still Using Insecure SHA-1 Certificates and Putting Users at Risk,” *Venafi Press Release*, 2017. .
[19] M. Stevens and D. Shumow, “Speeding up detection of SHA-1 collision attacks using unavoidable attack conditions.,” *USENIX Secur.*, vol. 2017, p. 173, 2017.
[20] 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.
[21] 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.
[22] X. Wang and H. Yu, “How to Break MD5 and Other Hash Functions,” *Adv. Cryptol. – EUROCRYPT 2005*, pp. 19–35, 2005.
[23] S. Verma and G. S. Prajapati, “Robustness and security enhancement of SHA with modified message digest and larger bit difference,” in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, pp. 1–5.
[24] K. kumar Raghuvanshi, P. Khurana, and P. Bindal, “Study and Comparative Analysis of Different Hash Algorithm,” *J. Eng. Comput. Appl. Sci.*, vol. 3, no. 9, pp. 1–3, 2014.
[25] J. Sharma and D. Koppad, “Low power and pipelined secure hashing algorithm-3(SHA-3),” in *2016 IEEE Annual India Conference (INDICON)*, 2016, vol. 3, pp. 1–5.
[26] A. Breust and F. Etcheverry, “Why not SHA-3?,” pp. 1–13, 2013.
[27] 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.
[28] P. Karpman, T. Peyrin, and M. Stevens, “Practical Free-Start Collision Attacks on 76-step SHA-1,” vol. 2012, 2015.
[29] G. Gupta and S. Sharma, “Enhanced SHA-192 algorithm with larger bit difference,” *Proc. - 2013 Int. Conf. Commun. Syst. Netw. Technol. CSNT 2013*, pp. 152–156, 2013.
[30] X. Xu, Q. Zhao, and C. Li, “Advanced framework for iterative hash functions,” *Proc. - 2012 Int. Conf. Comput. Sci. Electron. Eng. ICCSEE 2012*, vol. 2, pp. 599–602, 2012.
[31] C. C. G. San Jose, B. T. Tanguilig III, and B. D. Gerardo, “Enhanced SHA-1 on Parsing Method and Message Digest Formula,” pp. 1–9, 2015.
[32] L. Thulasmani and M. Madheswaran, “Security and Robustness Enhancement of Existing Hash Algorithm,” *2009 Int. Conf. Signal Process. Syst.*, pp. 253–257, 2009.
[33] M. Asgari Chenaghlu, S. Jamali, and N. Nikzad Khasmakhi, “A novel keyed parallel hashing scheme based on a new chaotic system,” *Chaos, Solitons & Fractals*, vol. 87, pp. 216–225, Jun. 2016.
[34] S. Deng, Y. Li, and D. Xiao, “Analysis and improvement of a chaos-based Hash function construction,” *Commun. Nonlinear Sci. Numer. Simul.*, vol. 15, no. 5, pp. 1338–1347, 2010.
[35] “Text Mechanic - Random String Generator Tool.” [Online]. Available: http://textmechanic.com/text-tools/randomization-tools/random-string-generator/.
[36] N. Smudge, “Hash Function Test Program,” 2008. [Online]. Available: http://nickmudge.info/?post=84.