VOL. 16, NO. 22, N0VEMBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

DESIGN AND DEVELOPMENT OF A COLLISION RESISTANT MODIFIED SHA1

Esmael V. Maliberan
Department of Information Technology, Surigao del Sur State University, Tandag City, Surigao del Sur, Philippines
E-Mail: malibs_28@yahoo.com

ABSTRACT

The Secure Hash Algorithm is best used in checking data integrity and authenticating digital media due to its fast hash calculating power. However, the SHA1 hashes are highly vulnerable to different attacks most especially collision attack because of the undersized hash length that compromises the integrity and confidentiality of data during transit. Presented in the current study is a new design to extend the hash size of SHA1 to secure a client-server communication through login authentication. Experimental results revealed that on five datasets that have collision in SHA1, none of them collided using the enhanced SHA1. Furthermore, the enhanced SHA1 is more resistant to brute force attack over the original SHA1 in terms of cracking time which is estimated to be sexagintillion years on a standard desktop PC. The results also presented that the produced hash codes of the enhanced SHA1 were resistant against a rainbow table and dictionary attack. By increasing the hash value of SHA1 from 160-bits to 1280-bits using AND and XOR operators, this paper has shown that execution of the enhanced SHA1 in digesting the passwords in login authentication delivers a safe way in a web transaction particularly in web-based environment.

Keywords: brute force attack, client-server communication, hashing, modified SHA1, rainbow table.

INTRODUCTION

Internet security is one of the interested topics in the field of Information Technology, most especially in a web-based environment [1], [2]. In the past few years, encryption and decryption are developing into main interest in data transmission [3]. One solution for preserving the transmitted data is a hash function. It is a portion of varied major Internet security claims [4]. It maintains integrity and confidentiality of the message and therefore these hashing algorithms are utilized mainly for file verification and user authentication [5], [6]. Hashing algorithms are additionally important elements in various security applications and practices [7]. SHA1 generates a 160-bit hash message [8], [9] based on principles of those utilized by Ronald L. Rivest who invented MD4 and MD5. It is the most commonly used hash algorithm, with its wide variety of applications [10], [11], [12] due to its adaptability, power [13], and agility [14]. The algorithm has been made as part of the capstone project of the government of United States and is usually applied to secure web transactions and other requests [15], [16]. While SHA1 is widely used, it does not seem to give adequate distributional avalanche effect of the input differences in the compression function [17], [10] by which this will lead to the probability of producing the same hash code of the two different input inside its compression function [18], [19]. Therefore, an improved diffusion needs to be configured to distribute the output at every stage inside its compression function [18], [20], [21]. The initial collision attack in SHA1 algorithm was attained by researchers, which produced two distinct PDF files. The study of [22] delivered another identical free-start pair for SHA-1 in a collision within its compression function. its undersize hash value would make it vulnerable to numerous attacks including a rainbow table, a dictionary, brute-force. Therefore, the SHA1 algorithm for client-server communication and login authentication is no longer safe. Some modifications and improvements were developed to address these problems [23], [24]. [25] Developed a new technique to improve the MD5 algorithm with the compression function of SHA1, which formed 256-bit hash code. On the other hand, [26] proposed a modification to the SHA-1 hash function. Rather than logical functions, the authors used the typical distributed pseudo-random function. These changes produced absolute hash values for original messages and contributed a hash value requirement to prevent brute attack. A number of hash functions were made, many of them were from the MD4 algorithm developed by Rivest [27]. Meanwhile, [28] argued that the enhancement of the SHA-1 was appropriate. According to the author, hackers and cryptanalysts found attacks on SHA-1 in 2005 which meant that the algorithm may not be safe enough to use. [29] the CPSO (Canonical Particle Swarm Optimization) method was used to boost the Stable Hash Algorithm-1. The idea was to forecast the block of control, which apprehend the password and provided the password stream with the same size of a log-list. [30] modified the method and formula to digest or decode the last SHA1 message by XORing, refining and expanding the mathematical formulation. The enhanced SHA-1 conserved its unique rounds covering of 80 rounds and an output of 160-bit hash value. [31] tested many collision attacks on the initial 80-step of the algorithm and proven a thorough adapted form of the procedure that weakened the probability of collision and elevated the time-complexity required to recognize an exponential factor of 2 in the collision. Presented in this paper is the modification of the algorithm by expanding its hash value from its original 160-bit to 1280-bit using the combined XOR and AND operator to prevent known attacks such as collision, dictionary, brute-force and rainbow table.

RESEARCH METHOD

Research Design

Simulations were done using Delphi programming language which was directly linked to MYSQL as database. XAMPP server 2.5 and MSQL server was used for storing the hash value of user passwords. All processes in this study were conducted on a personal computer running on a Celeron processor with 4Gb of memory and Windows 8, 64-bit operating system.

Five pairs of files that generated identical hash value in the original SHA1 were used for testing and verification. These file pairs are man.jpg and woman.jpg, text1.bin and text2.bin, set.bin and set1.bin, try.exe, and hi.exe, and f1.bak and f2.bak. The SHA1 hashing algorithm follows a Merkle-Damgard Construction. To address the flaws of SHA1, modification to the SHA1 output was made as shown in Figure-1.

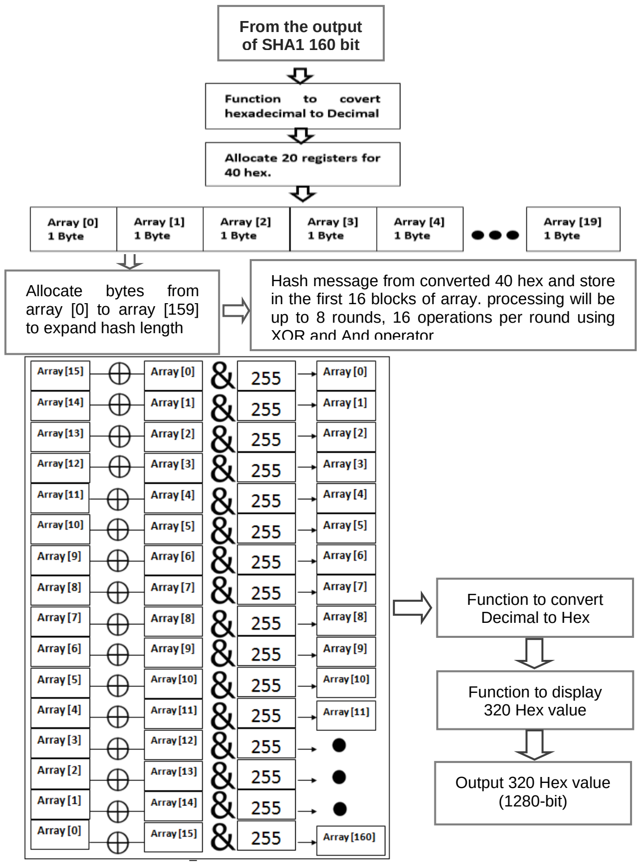

From the output of SHA1 160 bit
Function to covert hexadecimal to Decimal
Allocate 20 registers for 40 hex.
Array [0] 1 Byte
Array [1] 1 Byte
Array [2] 1 Byte
Array [3] 1 Byte
Array [4] 1 Byte
Array [19] 1 Byte
Allocate bytes from array [0] to array [159] to expand hash length
Hash message from converted 40 hex and store in the first 16 blocks of array. processing will be up to 8 rounds, 16 operations per round using XOR and And operator
Array [15] … Array [0]
Array [0] … Array [15]
255
Array [0] … Array [11]
Array [160]
Function to convert Decimal to Hex
Function to display 320 Hex value
Output 320 Hex value (1280-bit)

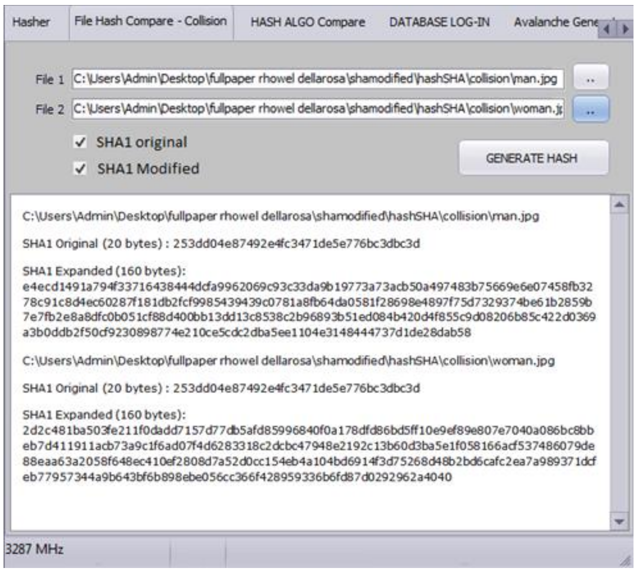
Figure-1. Architecture of the Modified SHA1.

Modification Process

The modification process of the study is shown in Figure-1. When a password is hashed using SHA1, its output will then be converted into a hex value encompassing 40 characters. There will be a function initiated to change this hex value into decimal and assign an array of 20 blocks containing 1 byte for each block. Two (2) characters will be assigned and allowed for every block. Additional function is made to assign an array of 160 blocks with a size of 1 Byte. It is designed to expand the hash size to 1280 bit as its proposed hash value. The message will be digested or hashed during this stage using XOR and AND operators and automatically stored it in an array of 20 blocks. In designing for modification, XOR was carefully considered due to its best bit shuffling properties [32], [33] which is proposed by many cyphers [34]. This operator is used in most of the cryptographic algorithms which has proven to be highly effective [35]. Moreover, the AND operator was added to 255 decimals since it is a required value of 1 byte to be stored in an array. The process will include up to eight rounds with 16 operations per round. Finally, a function is executed to convert the decimal value to hexadecimal and outputs the 320 hex hash value or 1280-bit hash.

Against Collision Attack

Five datasets that collided in SHA1 were used as input to the program to test whether it still collided in the enhanced SHA1. Every pair of files was uploaded into the simulation program that generates hash value using SHA1 and the enhanced SHA1 as shown in Figure-2.



File 1 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\man.jpg
File 2 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\woman.jp
✓ SHA1 original
✓ SHA1 Modified

```
C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\man.jpg

SHA1 Original (20 bytes) : 253dd04e87492e4fc3471de5e776bc3dbc3d

SHA1 Expanded (160 bytes):
e4ecd1491a794f33716438444dcfa9962069c93c33da9b19773a73acb50a497483b75669e6e07458fb32
78c91c8d4ec60287f181db2fcf9985439439c0781a8fb64da0581f28698e4897f75d7329374be61b2859b
7e7fb2e8a8dfc0b051cf88d400bb13dd13c8538c2b96893b51ed084b420d4f855c9d08206b85c422d0369
a3b0ddb2f50cf9230898774e210ce5cdc2dba5ee1104e3148444737d1de28dab58

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\woman.jpg

SHA1 Original (20 bytes) : 253dd04e87492e4fc3471de5e776bc3dbc3d

SHA1 Expanded (160 bytes):
2d2c481ba503fe211f0dadd7157d77db5afd85996840f0a178dfd86bd5ff10e9ef89e807e7040a086bc8bb
eb7d411911acb73a9c1f6ad07f4d6283318c2dcbc47948e2192c13b60d3ba5e1f058166acf537486079de
88eaa63a2058f648ec410ef2808d7a52d0cc154eb4a104bd6914f3d75268d48b2bd6cafc2ea7a989371dcf
eb77957344a9b643bf6b898ebe056cc366f428959336b6fd87d0292962a4040
```

Figure-2. Testing the enhanced SHA1 against collision attack.

Brute Force Attack

To measure the resilience of the modified against brute force attack, a password cracking tool (https://thycotic.force.com/support/s/article/Calculating-Password-Complexity) was used and the time needed to crack passwords were recorded. The possible combination based on character set is given by Equation 1. The cracking time, on the other hand, was computed using Equation 2.

$$\text{Possible combinations} = \text{possible number of characters}^{\text{Password length}} \qquad (1)$$

$$\text{Cracking time} = \left(\frac{\text{hash calculation per millisecond} * \text{possible password combination}}{2}\right) \qquad (2)$$

VOL. 16, NO. 22, N0VEMBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.

Dictionary Attack

To determine resilience against a dictionary attack, an online cracking tool called CrackStation (https://crackstation.net) that contains 1,493,677,782 words of possible passwords with equivalent hashes was used. The tool utilizes huge pre-computed lookup tables to break password hashes and these tables store a plot for that particular hash between the password hash and the right password. The hash codes are listed and indexed for searching the database immediately for a given confusion. If the message is found, the password can be obtained in a split second. It is applied only to messes without salt and supports current hash algorithms such as MD4, MD5, sha256, sha512, and others. Ten trials were done using the tool with different hash values generated from the enhanced SHA1 to guarantee precise resilience to a dictionary attack.

Rainbow Table Attack

To test the reliability of the modified SHA1 algorithm against a rainbow table attack, a pre-computed rainbow table was used. The table is made up of a database with a large number of hash function inputs and outputs. The aim of this test was to look for and compare a password with its hash value within the table. The Rainbow Crack tool was used to test a rainbow table attack (www.rainbowcrack.com).

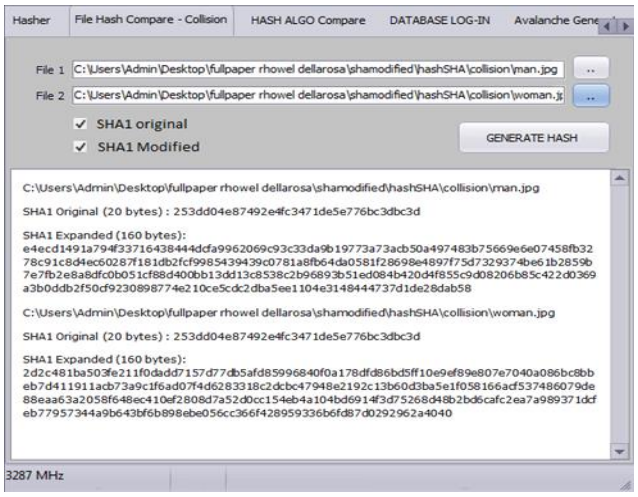
Application of the ENHANCED SHA1

A registration form and a login form were built for a user to register and communicate with the application server using Delphi Language. To use the system, a user was required to register a name, email address, and password which were then submitted to a server. The method is processed by a Delphi script, which then saves it to the database. Each time a user entered his email address and password, the modified SHA1 algorithm hashed the password into a 1280-bit hash length and compared its value to the stored hash value of the user's original password. The user's password was hashed into a 1280-bit length during registration, and stored in one password table in the server database. If the two passwords had the same hash values, the login authentication was verified and the user is given access to the server.

RESULTS AND ANALYSIS

Resistance to a Collision Attack

Five pairs of files were run and tested in a simulated program using the Delphi language. Shown in Figure-3 is the first pair of files run and tested using the program. Based on the experiment conducted, results show that on the first pair of images namely “man.jpg” and “woman.jpg”, the said files which collided using SHA1 produced different hash values when hashed by the modified SHA thus preventing collision attack. Shown in Figure-4 is the second pair of files run and tested using the simulation program.

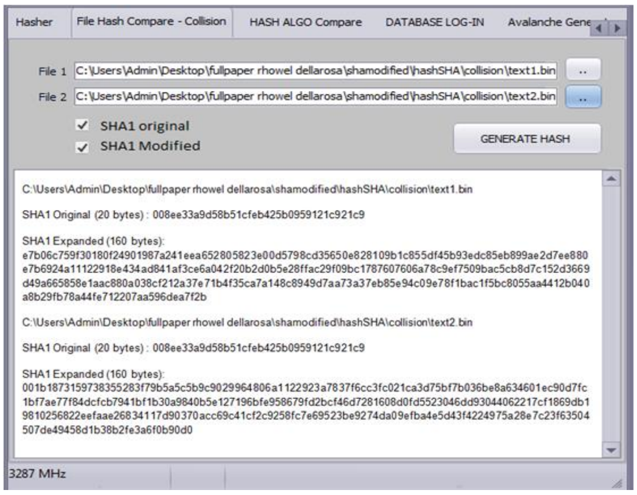


File 1 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\man.jpg
File 2 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\woman.j
✓ SHA1 original
✓ SHA1 Modified

```
C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\man.jpg

SHA1 Original (20 bytes) : 253dd04e87492e4fc3471de5e776bc3dbc3d

SHA1 Expanded (160 bytes):
e4ecd1491a794f33716438444dcfa9962069c93c33da9b19773a73acb50a497483b75669e6e07458fb32
78c91c8d4ec60287f181db2fcf9985439439c0781a8fb64da0581f28698e4897f75d7329374be61b2859b
7e7fb2e8a8dfc0b051cf88d400bb13dd13c8538c2b96893b51ed084b420d4f855c9d08206b85c422d0369
a3b0ddb2f50cf9230898774e210ce5cdc2dba5ee1104e3148444737d1de28dab58

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\woman.jpg

SHA1 Original (20 bytes) : 253dd04e87492e4fc3471de5e776bc3dbc3d

SHA1 Expanded (160 bytes):
2d2c481ba503fe211f0dadd7157d77db5afd85996840f0a178dfd86bd5ff10e9ef89e807e7040a086bc8bb
eb7d411911acb73a9c1f6ad07f4d6283318c2dcbc47948e2192c13b60d3ba5e1f058166acf537486079de
88eaa63a2058f648ec410ef2808d7a52d0cc154eb4a104bd6914f3d75268d48b2bd6cafc2ea7a989371dcf
eb77957344a9b643bf6b898ebe056cc366f428959336b6fd87d0292962a4040
```

3287 MHz

Figure-3. Result for testing a collision attack.



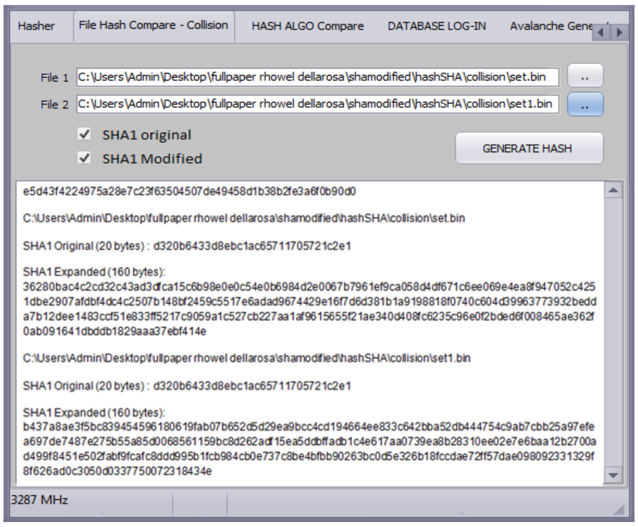
File 1 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\text1.bin
File 2 C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\text2.bin
✓ SHA1 original
✓ SHA1 Modified

```
C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\text1.bin

SHA1 Original (20 bytes) : 008ee33a9d58b51cfeb425b0959121c921c9

SHA1 Expanded (160 bytes):
e7b06c759f30180f24901987a241eea652805823e00d5798cd35650e828109b1c855df45b93edc85eb899ae2d7ee880
e7b6924a11122918e434ad841af3ce6a042f20b2d0b5e28ffac29f09bc1787607606a78c9ef7509bac5cb8d7c152d3669
d49a665858e1aac880a038cf212a37e71b4f35ca7a148c8949d7aa73a37eb85e94c09e78f1bac1f5bc8055aa4412b040
a8b29fb78a44fe712207aa596dea7f2b

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\text2.bin

SHA1 Original (20 bytes) : 008ee33a9d58b51cfeb425b0959121c921c9

SHA1 Expanded (160 bytes):
001b18731597383552 83f79b5a5c5b9c9029964806a1122923a7837f6cc3fc021ca3d75bf7b036be8a634601ec90d7fc
1bf7ae77f84dcfcb7941bf1b30a9840b5e127196bfe958679fd2bcf46d7281608d0fd5523046dd93044062217cf1869db1
9810256822eefaae26834117d90370acc69c41cf2c9258fc7e69523be9274da09efba4e5d43f4224975a28e7c23f63504
507de49458d1b38b2fe3a6f0b90d0
```

3287 MHz

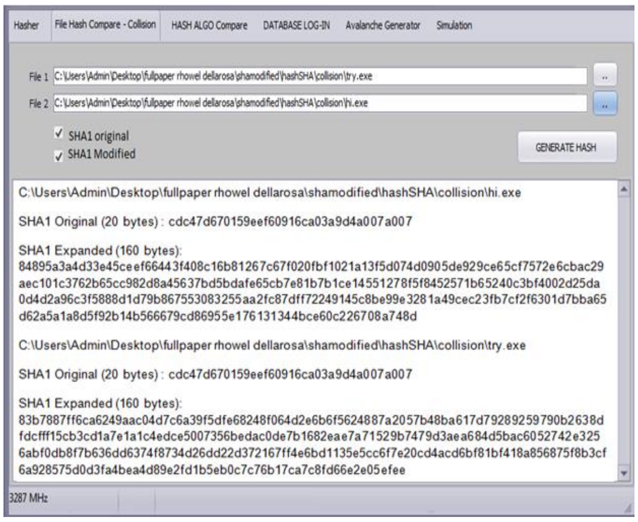
Figure-4. Result in testing collision resistance on text1.bin and text2.bin.

On the second pair of binary files namely “text1.bin” and “text2.bin,” results indicate that the said files generated different hash code using enhanced SHA1 to prevent collision attack. Unlike SHA1, the modified SHA1 is collision-resistant. Shown in Figure-5 is the result of the simulation on the third pair of files. It can be observed that the results showed that on the third pair of binary files namely “set.bin” and “set1.bin”, the system produced different hash values using the modified SHA1. Another test was done on the two different executable files namely the “try.exe” and “hi.exe.” as shown in Figure-6.



```
File 1  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\set.bin
File 2  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\set1.bin
✓ SHA1 original
✓ SHA1 Modified

e5d43f4224975a28e7c23f63504507de49458d1b38b2fe3a6f0b90d0

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\set.bin

SHA1 Original (20 bytes) : d320b6433d8ebc1ac65711705721c2e1

SHA1 Expanded (160 bytes):
36280bac4c2cd32c43ad3dfca15c6b98e0e0c54e0b6984d2e0067b7961ef9ca058d4df671c6ee069e4ea8f947052c425
1dbe2907afdbf4dc4c2507b148bf2459c5517e6adad9674429e16f7d6d381b1a9198818f0740c604d39963773932bedd
a7b12dee1483ccf51e833ff5217c9059a1c527cb227aa1af9615655f21ae340d408fc6235c96e0f2bded6f008465ae362f
0ab091641dbddb1829aaa37ebf414e

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\set1.bin

SHA1 Original (20 bytes) : d320b6433d8ebc1ac65711705721c2e1

SHA1 Expanded (160 bytes):
b437a8ae3f5bc839454596180619fab07b652d5d29ea9bcc4cd194664ee833c642bba52db444754c9ab7cbb25a97efe
a697de7487e275b55a85d0068561159bc8d262adf15ea5ddbffadb1c4e617aa0739ea8b28310ee02e7e6baa12b2700a
d499f8451e502fabf9fcafc8ddd995b1fcb984cb0e737c8be4bfbb90263bc0d5e326b18fccdae72ff57dae098092331329f
8f626ad0c3050d0337750072318434e
```

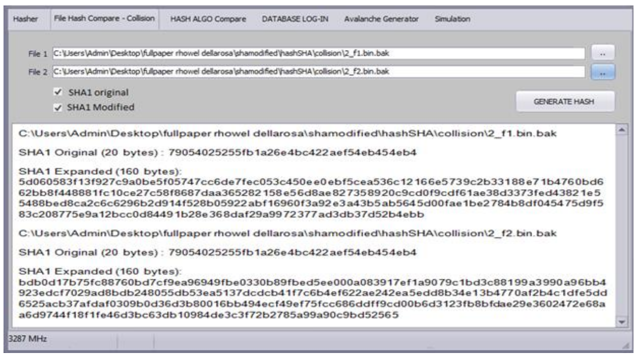
Figure-5. Result on collision attack on set.bin and set1.bin.



```
File 1  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\try.exe
File 2  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\hi.exe
✓ SHA1 original
✓ SHA1 Modified

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\hi.exe

SHA1 Original (20 bytes) : cdc47d670159eef60916ca03a9d4a007a007

SHA1 Expanded (160 bytes):
84895a3a4d33e45ceef66443f408c16b81267c67f020fbf1021a13f5d074d0905de929ce65cf7572e6cbac29
aec101c3762b65cc982d8a45637bd5bdafe65cb7e81b7b1ce14551278f5f8452571b65240c3bf4002d25da
0d4d2a96c3f5888d1d79b867553083255aa2fc87dff72249145c8be99e3281a49cec23fb7cf2f6301d7bba65
d62a5a1a8d5f92b14b566679cd86955e176131344bce60c226708a748d

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\try.exe

SHA1 Original (20 bytes) : cdc47d670159eef60916ca03a9d4a007a007

SHA1 Expanded (160 bytes):
83b7887ff6ca6249aac04d7c6a39f5dfe68248f064d2e6b6f5624887a2057b48ba617d79289259790b2638d
fdcfff15cb3cd1a7e1a1c4edce5007356bedac0de7b1682eae7a71529b7479d3aea684d5bac6052742e325
6abf0db8f7b636dd6374f8734d26dd22d372167ff4e6bd1135e5cc6f7e20cd4acd6bf81bf418a856875f8b3cf
6a928575d0d3fa4bea4d89e2fd1b5eb0c7c76b17ca7c8fd66e2e05efee
```

Figure-6. Result on collision attack on try.exe and hi.exe.

It can be noted that the results showed that the two executable files collided in SHA1 while it did not collide using the modified SHA1. The last test was done on two different backup files namely "f1.bak" and "f2.bak". The results are shown in Figure-7. Consistent with the previous results, the system for modified SHA1 produced different hash values for the two files. Shown in Table-1 is the summary of the test results for the collision attack for the five pairs of files that were used during the experimentation.



```
File 1  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\2_f1.bin.bak
File 2  C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\2_f2.bin.bak
✓ SHA1 original
✓ SHA1 Modified

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\2_f1.bin.bak

SHA1 Original (20 bytes) : 79054025255fb1a26e4bc422aef54eb454eb4

SHA1 Expanded (160 bytes):
5d060583f13f927c9a0be5f05747cc6de7fec053c450ee0ebf5cea536c12166e5739c2b33188e71b4760bd6
62bb8f448881fc10ce27c58f8687daa365282158e56d8ae827358920c9cd0f9cdf61ae38d3373fed43821e5
5488bed8ca2c6c6296b2d914f528b05922abf16960f3a92e3a43b5ab5645d00fae1be2784b8df045475d9f5
83c208775e9a12bcc0d84491b28e368daf29a9972377ad3db37d52b4ebb

C:\Users\Admin\Desktop\fullpaper rhowel dellarosa\shamodified\hashSHA\collision\2_f2.bin.bak

SHA1 Original (20 bytes) : 79054025255fb1a26e4bc422aef54eb454eb4

SHA1 Expanded (160 bytes):
bdb0d17b75fc88760bd7cf9ea96949fbe0330b89fbed5ee000a083917ef1a9079c1bd3c88199a3990a96bb4
923edcf7029ad8bdb248055db53ea5137dcdcb41f7c6b4ef622ae242ea5edd8b34e13b4770af2b4c1dfe5dd
6525acb37afdaf0309b0d36d3b80016bb494ecf49ef75fcc686ddff9cd00b6d3123fb8bfdae29e3602472e68a
a6d9744f18f1fe46d3bc63db10984de3c3f72b2785a99a90c9bd52565
```

Figure-7. Result in testing collision attack.

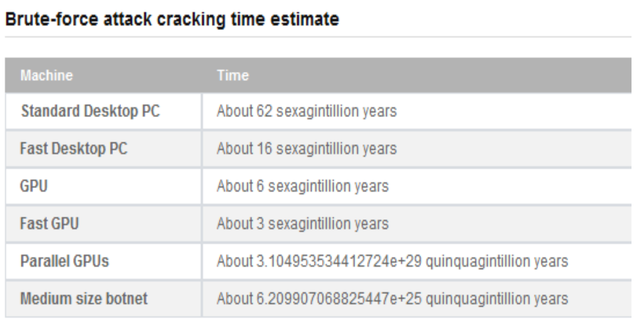
Table-1. Experimentation result for collision attack.

Pair of Files	SHA1 Algorithm	
	Original	**Modified**
man.jpg and woman.jpg	Successful	Failed
text1.bin and text2.bin	Successful	Failed
set.bin and set1.bin	Successful	Failed
try.exe and hi.exe	Successful	Failed
f1.bak and f2,bak	Successful	Failed

It should be interpreted from Table-1 that when testing each pair of files for collision resistance, the same hash values were produced by the original SHA1 while this was not the case for the enhanced SHA1 algorithm. The enhanced SHA1 has thus been able to resolve the collision attack.

Resistance to a Bruteforce Attack

Resistance to brute force attack was done using the password checker online tool. The UI for this tool is shown in Figure-8 which also gives details of the program.



Brute-force attack cracking time estimate

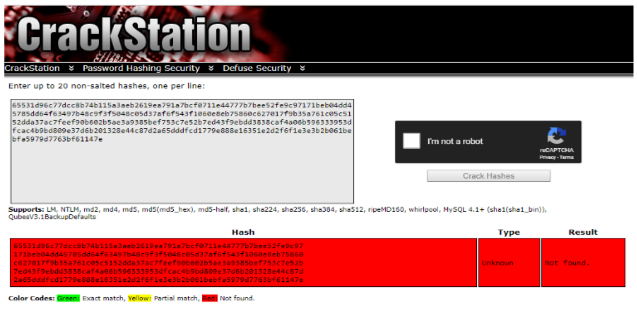
Machine	Time
Standard Desktop PC	About 62 sexagintillion years
Fast Desktop PC	About 16 sexagintillion years
GPU	About 6 sexagintillion years
Fast GPU	About 3 sexagintillion years
Parallel GPUs	About 3.104953534412724e+29 quinquagintillion years
Medium size botnet	About 6.209907068825447e+25 quinquagintillion years

Figure-8. UI for password checker online to crack the hash code.

The output hash value of the modified SHA1 algorithm was tested; it was revealed that it would take about 62 sexagintillion years to crack. This length of time renders the modified SHA1 virtually impossible to crack using currently-available cracking tools, unlike the original SHA1.

Resistance to Dictionary Attack

To assess how robust, the enhanced SHA1 is against a dictionary attack, CrackStation was used. The UI for the system, which also shows a result for the dictionary attack, is shown in Figure-9.



Enter up to 20 non-salted hashes, one per line:

```
65531d96c77dcc8b74b115a3aeb2619ea791a7bcf0711e44777b7bee52fe9c97171beb04dd4
5785dd64f63497b48c9f3f5048c05d37af6f543f1060e8eb75860c627017f9b35a761c05c51
52dda37ac7feef90b602b5ae3a9385bef753c7e52b7ed43f9ebdd3838caf4a06b596333953d
fcac4b9bd809e37d6b201328e44c87d2a65dddfcd1779e888e16351e2d2f6f1e3e3b2b061be
bfa5979d7763bf61147e
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
65531d96c77dcc8b74b115a3aeb2619ea791a7bcf0711e44777b7bee52fe9c97 171beb04dd45785dd64f63497b48c9f3f5048c05d37af6f543f1060e8eb75860 c627017f9b35a761c05c5152dda37ac7feef90b602b5ae3a9385bef753c7e52b 7ed43f9ebdd3838caf4a06b596333953dfcac4b9bd809e37d6b201328e44c87d 2a65dddfcd1779e888e16351e2d2f6f1e3e3b2b061bebfa5979d7763bf61147e	Unknown	Not found.

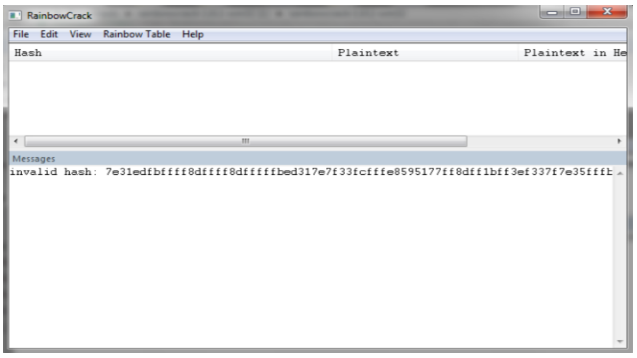
Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Figure-9. Crack Station UI.

For the input of the plaintext "admin", results showed that the tool was unable to generate a plaintext hash code generated by the enhanced SHA1 in its database. The tool was unable to recognize the hash format, and its type was either unknown or unidentified in its database. This is due to the fact that the method only works with well-known hashing algorithms like MD5, SHA256, SHA512, and MD160.

Resistance to Rainbow Table Attack

As shown in Figure-10, another attack was used to check the reliability of the modified SHA1 algorithm using a rainbow table attack.



```
invalid hash: 7e31edfbffff8dffff8dfffffbed317e7f33fcfffe8595177ff8dff1bff3ef337f7e35fffb
```

Figure-10. Cracking the hash code using a rainbow table attack.

Results indicated that the tool failed to crack the "admin" hash value that is 1280-bit created from the enhanced SHA1. This can be due to the fact that only hashes for known hash algorithms such as MD5, SHA2, SHA256, and SHA512 are in the database. Moreover, the same hash codes are used during a dictionary attack to verify that the modified SHA1 is resilient to a rainbow table attack.

Comparison of the Results of SHA Family on Collision Attack

A comparison of the different SHA family along with the proposed enhancement is significant because it allows other readers and researches to appreciate the features of the proposed study that is not present in the existing SHA family. Table-2 compares improved SHA1 to SHA1, SHA256, SHA384, and SHA512 in terms of computing pre-images attack, second pre-image attack, and heavy collision attack.

Table-2. Comparison Results of different SHA algorithms and the enhanced SHA1 on Collision Attack.

Name of the attack	SHA1	SHA256	SHA384	SHA 512	This study
Pre-image attack	2^{160}	2^{256}	2^{384}	2^{512}	2^{1280}
Second Pre-image attack	2^{160}	2^{256}	2^{384}	2^{512}	2^{1280}
Strong Collision attack	2^{80}	2^{128}	2^{192}	2^{256}	2^{640}

The pre-image for an n-bit hash code is determined by formula 2n, where n is the number of bits generated, according to [36]. Strong collision is also measured by the formula 2n/2, where n is the number of bits generated by a hash algorithm. This implies that the longer the hash value, the more operations and permutations the attacker would have to run in order to carry out such attacks.

CONCLUSIONS

Based on the results of the study, the following conclusions are drawn, 1. The modified SHA1 algorithm was evaluated to be 100% resistant to the collision attack. 2. The modified SHA1 algorithm exhibits a very high level of resilience against brute-force attack based on the approximately 1.6907x10^{471}years it would take to crack generated hash values. 3 The modified SHA1 algorithm exhibits 100% resilience against a dictionary attack and 100% resilience against a rainbow table attack. The conclusions above show that modification of SHA1 through the expansion of its hash value from 160 bits to 1280 bits is effective in addressing the collision problem of SHA1. Furthermore, the modification was able to generate hash values that were resilient against the common forms of security attacks. Having produced a more secured and modified SHA1, it can be said that this study has successfully achieved its objectives.

ACKNOWLEDGEMENT

The author acknowledges the support of the Research and Development office of Surigao del Sur State University.

REFERENCES

[1] G. Raj, R. Kesireddi and S. Gupta. 2015. Enhancement of Security Mechanism for Confidential Data using AES-128, 192 and 256bit Encryption in Cloud, 1st International Conference on Next Generation Computing Technologies (NGCT-2015), 374-378.

[2] A. Arora, A. Rastogi, A. Khanna and A. Agarwal. 2017. Cloud Security Ecosystem for Data Security and Privacy, 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence. 288-292.

[3] M. Ramesha, S.B. Sridhara, A.B. Gowda, N. Anughna and G. Bharathi. 2020. Design and Development of Low Power BTED Cryptography Algorithm on FPGA, International Journal of Advanced Trends in Computer Science and Engineering. 9(4): 4359-4362.

[4] D. Morres. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Federal Information Processing Standards Publication.

[5] L. Zhong, W. Wan and D. Kong. 2016. Java web login authentication based on improved md5 algorithm. 131-135.

[6] X. Zheng and J. Jin. 2012. Research for the Application and Safety of MD5 Algorithm in Password Authentication, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012). 2216-2219.

[7] P. Walia and V. Thapar. 2015. Implementation of New Modified MD5-512 bit Algorithm for Cryptography, International Journal of Innovative Research in Advanced Engineering (IJIRAE). 1(6): 87-97.

[8] NIST. FIPS. 1995. Secure Hash Standard**,** FIPS PUB. 180-1, 17.

[9] Q. Dang. 2015. Secure Hash Standard, Gaithersburg. 1-36.

[10] N. Kishore and B. Kapoor. 2016. Attacks on and advances in secure hash algorithms, IAENG Int. J. Comput. Sci. 43(3): 326-335.

[11] S. Rao. 2015. Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity, Int. J. Comput. Appl. 130(8): 25-27.

[12] R. Karthik and A.K. Parvathy. 2017. Non-convex Economic Load Dispatch using Cuckoo Search Algorithm, IndonesJ Electr. Eng. ComSci. 5(1): 48-57.

[13] K. Saravanan and A. Senthilkumar. 2013. Theoretical Survey on Secure Hash Functions and issues, Int. J. Eng. Res. Technol. 2(10): 1150-1153.

[14] P. Garg and N. Tiwari. 2012. Performance Analysis of SHA Algorithms (SHA-1 and SHA-192): A Review 2(3): 130-132.

[15] S. Song. 2012. A Study on Area-Efficient Design of Unified MD5 and HAS-160 Hash Algorithms, The Journal of the Korean Institute of Information and Communication Engineering. 5(16): 1015-1022.

[16] M. Chouhan and V. Kapoor. 2015. A new security system using ECC AND MD5, International Journal of Engineering Research & Technology. 1798-1801.

[17] X. Wang, Y. Yin and H. Yu. 2005. Finding Collisions in the Full SHA-1, Adv. Cryptol. - CRYPTO 2005. 17-36.

[18] A. Kumarkasgar, J. Agrawal and S. Shahu. 2012. New modified 256-bit MD5 Algorithm with SHA Compression Function, Int. J. Comput. Appl, 42(12): 47-51.

[19] P. Karpman, T. Peyrin and M. Stevens. 2015. Practical Free-Start Collision Attacks on 76-step SHA-1, 2: 1-22.

[20] G. Gupta and S. Sharma. 2013. Enhanced SHA-192 algorithm with a larger bit difference, Proc. - 2013 Int. Conf. Commun. Syst. Netw. Technol. CSNT. 152-156.

[21] X. Xu, Q. Zhaoand C. Li. 2012. Advanced framework for iterative hash functions, Proc. - 2012 Int. Conf. Comput. Sci. Electron. Eng. ICCSEE. 2: 599-602.

[22] M. Stevens, P. Karpman and T. Peyrin. 2016. Freestart Collision for Full SHA-1, Annual International Conference on the Theory and

Applications of Cryptographic Techniques. 9665: 459-483.

[23] A. Bhandari. 2017. Enhancement of MD5 Algorithm for Secured Web Development, Journal of Software, 12(4): 240-252.

[24] Y. Sasaki. 2015. Improved Single-Key Distinguisher on HMAC-MD5 and Key Recovery Attacks on Sandwich-MAC-MD5 and MD5-MAC, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences.1: 26-38.

[25] A. Kasgar, J. Agrawal and S. Sahu. 2012. New Modified 256-bit MD5 Algorithm with SHA Compression Function, International Journal of Computer Applications. 42(12): 47-51.

[26] Asbduvaliyev, S. Lee and Y. Lee. 2016. Modified SHA1 hash function (mSHA1).

[27] R. Rivest. 1991. The MD4 message-digest algorithm. CRYPTO'90, LNCS. 303–311.

[28] H. Tiware and K. Asawa. 2013. Enhancing the Security Level of SHA-1 by Replacing the MD Paradigm, Journal of Computing and Information Technology -CIT 21, 4: 223–233.

[29] M. Alam and S. Ray. 2013. Design of an Intelligent SHA-1 Based Cryptographic System: A CPSO Based Approach, International Journal of Network Security.15(6): 465-470.

[30] San Jose, B. Gerardo and B. Tanguilig. 2015. 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, Philippines. 1-9.

[31] R. Siddhartha. 2015. Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity, International Journal of Computer Applications, 8(130).

[32] Dobre. 2013. Hash Function XOR. Information Security.

[33] Vandierndonck and K. De Bosschere. 2005. XOR-based Hash Function, IEEE Transactions on Computers, 54(7): 800-812.

[34] K. Kiran and R. Shantharama. 2020. FPGA Implementation of Simple Encryption Scheme for Resource-Constrained Devices, International Journal of Advanced Trends in Computer Science and Engineering. 9(4): 5631-5639.

[35] S. Nagesh, K. Anil Kumar, S. Suchitra and S. Abhishek. 2020. Data Security in Cloud Environment Based on Comparative Performance Evaluation of Cryptographic Algorithms. International Journal of Advanced Trends in Computer Science and Engineering. 9(4): 4989-4997.

[36] Gordon. 2016. Properties of Hash Functions, Sirindhorn International Institute of Technology, Thailand: Thammasat University.