International Journal of Communication Networks and Information Security (IJCNIS) Vol. 11, No. 1, April 2019

Modified SHA1: A Hashing Solution to Secure Web Applications through Login Authentication

Esmael V. Maliberan

Graduate Studies, Surigao del Sur State University, Philippines

***Abstract*:** The modified SHA1 algorithm has been developed by expanding its hash value up to 1280 bits from the original size of 160 bit. This was done by allocating 32 buffer registers for variables A, B, C and D at 5 bytes each. The expansion was done by generating 4 buffer registers in every round inside the compression function for 8 times. Findings revealed that the hash value of the modified algorithm was not cracked or hacked during the experiment and testing using powerful online cracking tool, brute force and rainbow table such as Cracking Station and Rainbow Crack and bruteforcer which are available online thus improved its security level compared to the original SHA1.

***Keywords*:** SHA1, hashing, client-server communication, modified SHA1, hacking, brute force, rainbow table

1. Introduction

Cryptography and Network Security are concepts to secure client-server communication over wireless network. Currently, Internet security has been an interest in the area of Information Technology particularly in cloud environment [1, 2,]. According to [23], in order to secure data in a cloud environment, a fully homomorphic encryption (FHE) scheme with improved efficiency can be implemented. Cryptographic hash function is one of the solutions to protect data being transferred. It is a component for various signification internet security applications [3]. Moreover, it accepts any length of messages and parsed it into fixed length value, which is called information abstraction. Hashing cannot be reversed unlike encryption. It has main usage in sustaining message integrity and confidentiality and because of this, hash algorithms are used especially in login authentication and file verification [4, 5]. In addition, hashing algorithms are essential elements in numerous security applications and practices [6]. SHA-1 generates a hash message based on principles related to those used by Ronald L. Rivest who designed MD4 and MD5 algorithms. The algorithm was made as a portion of the U.S. Government's Capstone project and it is widely used in securing client-server communications and a variety of applications [7, 8]. Secure Hash Algorithm (SHA 1), is a cryptographic hashing algorithm utilized to verify the authenticity and integrity data. Differences of this algorithm are usually applied by SSL certificate authorities to sign certificates and digital signatures. This hash algorithm guarantees that the website's data is not tampered or altered. It generates a fixed 160 bit unique hash codes from any file size. Based from these hash values, it can be verified whether or not the file has been modified by comparing the hash value generated from the hash value stored in the database [24].

But, SHA1 is not perfect due to its short hash value. It can easily be cracked using brute force and rainbow table. Furthermore, researchers have attained the first collision attack against the SHA-1 hash function, generating two different PDF files. The research study conducted by [9] presented a specific freestart identical pair for SHA-1, *i.e.* a collision within its compression function. This was the first appropriate break of the SHA-1, extending all 80 out of 80 steps. This attack was performed for only 10 days of computation on a 64-GPU. Thus, SHA1 algorithm is not anymore safe in login authentication and data transfer. For this reason, there were enhancements and modifications being developed in the algorithm in order to solve these issues [10, 11]. [12] proposed a new approach to enhance MD5 algorithm combined with SHA compression function that produced a 256-bit hash code. This enhancement was used to prevent birthday attacks, rainbow table and brute force attacks. The method was to extend the hash length up to 256–bit from its original size of 128-bit. The block size was expanded inside the compression function. The proposed modification was applied in data reliability and signing applications. Meanwhile [13] proposed a modification to SHA-1 hash function. For their scheme, the authors utilized the usually dispersed pseudo random function instead of logical functions. This change generated an exclusive hash codes for distinctive messages and provided brute force-resistance requirement to hash function. Numerous hash functions have been proposed, majority of them were derived from MD4 hash function. The MD4 hash function was developed by Rivest [14].

2. Related Work

A new secure Hash algorithm, SHA-192 is presented in [15]. The novel SHA-1 established by NIST generated 160 bit message digest whereas the proposed SHA-192 generated a 192 output hash size. The authors made modifications to the original function and examined that SHA-192 to be much improved than the present SHA-1 hashing algorithm with regards to the number of brute force attack. However, the time performance of the proposed SHA-192 was slower compared to the original one since it needs to generate a 192 bit of hash value. A digital signature consisting mathematical calculations that demonstrates the authenticity of a message was also proposed as an enhancement to the original SHA1 [16].

Meanwhile [17] argued the necessity to modify the SHA-1. According to the author, a hash function in one of them most important cryptographic approaches which are usually used in many applications. However, it was found out that these hash functions that followed the Merkle-Damgard Construction have shown weaknesses particularly on the compression function thus vulnerable to generic attacks and collision attacks. Therefore, the proposed study based it on the design principle of SHA-1 and is patterned on dither construction. The compression function part consists of three inputs and produces a fix output of 160-bit length. Findings revealed that Dither construction have shown a strong resistance against different cryptanalytic attacks.

A Canonical Particle Swarm Optimization approach to improve the SHA1 is presented in [18]. The idea consisted of predicting control block which captured the plaintext from the user and provided a log-list of the same size with the plaintext stream. The forecasting scheme does obstruct the CPU utilization a bit but the author was convinced that the innovative scheme will produce a new setting in crafting a cryptographic hash function.

Moreover, a new improved MD5 Algorithm combined with SHA Compression Function that produced a 256-bit hash code is presented in [19]. This enhancement was used to prevent brute force and rainbow table and birthday attacks. Their method was to expand the hash size to 256–bit from its original size of 128-bit by expanding the compression function block size which will be applied in data reliability and signing applications. Complicated message enhancement approaches were used to attain the essential situations on the chaining variables and the message bits. Findings revealed that it was resistant to local collision and differential attack.

In addition, [20] made enhancement on SHA1 particulary on the method and the formula of parsing or digesting the final message through shifting, xoring and improved mathematical formula. The improved SHA-1 maintained its original rounds which consisted of 80 rounds and message digest output of 160 bit.

An advanced SHA-1 Algorithm ensuring stronger data integrity is presented in [21]. The study evaluated many collision attacks on the earliest complete 80-step SHA-1 and presented an innovative enhanced edition of the algorithm that decreased the likelihood of collision and augmented the theoretical lower bound of the time complexity needed to discover the collision by an exponential factor of 2.

In 1993, the original SHA1 algorithm was invented by The National Security Agency (NSA) and was announced by the National Institute of Standards and Technology (NIST) as Federal Information Processing Standard (FIPS) 180, letting its application and usage in government projects. After the publication of SHA1, NSA abolished the standard and included minor modification to the hash function. The NSA confirmed that the alteration deal with an error in the original algorithm that reduced its cryptographic security. However, NSA was unable to give details regarding the error, which leaded in utilizing and consuming a lot of time scrutinizing the algorithm. Thus, SHA1 algorithm is still believed to be safe to use in hashing applications because no flaws was found in it.

The NIST published FIPS 180-2 sometime in 2001, made several enhancements which are now defined as SHA-256, SHA-384, and SHA-512. The name of each algorithm was due to the hash sizes each one produced. The creations of these new algorithms were variants of the original SHA-1 algorithm and were sufficiently new that their security remained open.

Properties of a hash algorithm. Pre-image resistant: for any known h, it is infeasible to find y such that $H(y)=h$ while in second pre-image resistant, for any given x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$. For strong collision resistant, it is infeasible to find pair (x,y) such that $H(x) = H(y)$.

Table 1 Summary of different hashing algorithms

Name	Input block size	Message limit (bits)	Hash code size (bits)
MD5	512	2^{64}	128
SHA-1	512	2^{64}	160
SHA-256	512	2^{64}	256
SHA-384	1024	2^{128}	384
SHA-512	1024	2^{128}	512

These properties of a hash algorithm must be satisfied by the proposed algorithm. According to [22], for a given *n*-bit hash code, pre-image is calculated by the formula 2^n where n is the number of bits output. Moreover, strong collision is calculated by the formula $2^{n/2}$ where n is the number of bits output of a hash algorithm.

Table 2 Comparison of the modified MD5 to the MD5 in calculating pre-images attack, second pre-image attack and strong collision attack

Name of Attacks	MD5 (128-bit output)	Modified MD5 (1280-bit output)
Pre-image attack	2^{128}	2^{1280}
Second Pre-image attack	2^{128}	2^{1280}
Strong Collision attack	2^{64}	2^{640}

As shown in Table 2, it would take 2^{128} permutations before a pre-image attack is calculated using MD5 while it would take 2^{1280} permutations for the proposed algorithm. Second pre-image attack would take 2^{1280} permutations and strong collision attack would take 2^{640} permutations for the proposed algorithm. This means that the longer the length of the hash code output is, the more operations and permutations the attacker needs to perform to these attacks.

Table 3 Usage of the properties of a Hash Algorithm

Applications	Pre-image Resistant	Second Pre-image Resistant	Collision Resistant
Hash + digital signature	Yes	Yes	Yes
Intrusion detection and virus detection		Yes	
Hash + symmetric encryption			
One-way password file	Yes		
MAC	Yes	Yes	Yes

3. System Architecture, Materials and Method

In this section, the approach in expanding the hash size of the modified SHA1 is described. Materials which include both the software and hardware are identified and the procedure of the different phases is defined.

3. 1. Expanding the hash value of SHA1

First the message is padded (extended) so that its length (in bits) is congruent to 448, modulo 512. Next, 64 bits is appended to the message so that it will be equal to 512 bit. Then the buffer registers A, B, C, D at 40 bits each is initialized to a certain constant before it will be processed in 20 word block. The message will be processed from round 1 to round 4 and every round consists of 20 steps at 5 bytes of the message per step. If the message is lesser than 512 bit, only 1 cycle of rounds will be needed before an output of 160 bit SHA1 is produced. To expand the hash value to 1280 bit, 32 buffer registers (buff [0] to buff [31]) is allocated containing 5 Bytes each. The original output of SHA1 generated from buffer A, B, C, and D was used to process for 8 rounds inside the compression function (1-4 rounds). 4 buffer registers were created in every round which generated a total of 160 Bytes or 1280 bits hash value as shown in Figure 1 which is the system architecture of the study.

3.2. Software

The modified SHA1 algorithm which has a hash size of 1280 bit was used for Login authentication in various web applications. One of the software used to implement this was MYSQL database. PHP Programming Language. PHP (Hypertext Pre-processor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Other software used include XAMPP server and MSQL server.

3.3 Hardware

The hardware that was used in this study was Intel Core i5 7100 3.9GHz, 8G Ram DDR3, 500 HDD Hard Drive.

3.4 Experimental design

The modified SHA1 hashed the password into 1280 bit created by the user during registration. Using the PHP scripting language, it stored the hashed password in the user's table of MySQL database running in the server. Every time users fill the basic information such as username and password and clicks on the log in button, the information is submitted to the server in a plain text. Using the modified SHA1 algorithm, the script in the server digested or hashed the password inputted by the user into 1280 bits size. Moreover, there was a function that compared the generated hash code of the password from the users and the hash codes stored in the table of MySQL database. If there is a match in the in the hash values converted by the modified SHA1 in the database, then the login authentication is successful and the user can access the system.

3.5 Procedures for the different phases

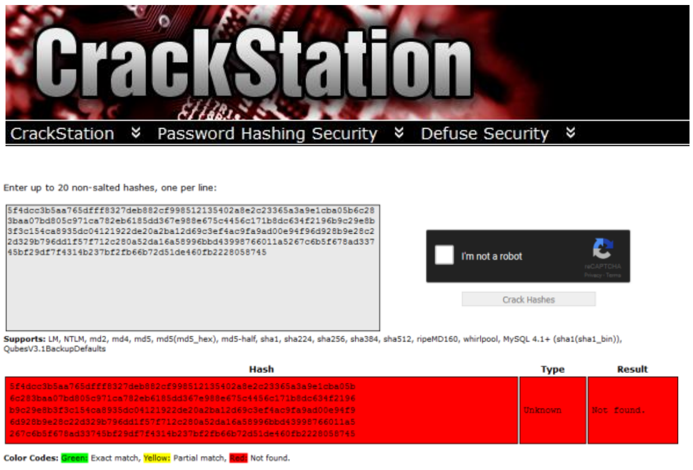
Phase 1. Expanding hash length from 128 bit to 1280 bit
The expansion was done by allocating 32 buffer registers at 5 Bytes each. The process was done at 8 runs in the compression function of SHA1 to produce 1280 bits of hash value.
Phase 2. Designing a Login procedure for user's authentication
In this phase, a log in form was created using PHP scripting language for a user who wanted to register and communicate with the application server. Every time a user fills the basic information such as username and password and clicks on the log in button, the information is submitted to the server. When a user will register or create an account, a script in PHP will process the procedure and store it in the database.
Phase 3. Applying the modified SHA1 in securing a client server communication
Every time a user enters his username and password, the modified sha1 algorithms hashes the password into 1280 bit hash length and compare its value from the stored hash value of the user's original password. During registration, user's password is being hashed into 1280 bit length and will be stored in one mypassword table in the server's database. If the hash values of the two passwords are the same, then the login authentication is verified and the user will be able to access the server.

3.6 Evaluation

During the evaluation, the researcher compared the original SHA1 algorithm to the modified SHA1 algorithm with respect to the length of their respective hash values. By applying the modified SHA1 in the system or any web application during login authentication will be impossible for the attacker to use a rainbow table in order to lookup for the possible string of the 1280 bit hash value because its length is eight (8) times the length of the original SHA1. In most cases, attackers only have all the possible combinations of a 128 bit hash value generated from SHA1 in their databases

4. Testing the reliability of the modified SHA1

To conduct a comparison in terms of the security level of SHA1 and the modified SHA1, tools for online password cracker, brute force and rainbow table were used to simulate the attack using the two algorithms. Since the modified SHA1 generated 1280 bit hash value, these attacks did not work because the hash value is not found in the databases of the existing hash algorithms like SHA1, SHA256, SHA 512, MD5, MD4, RIPEMD320, etc. thus improved the security level of the modified SHA1.
Shown in Figure 2, the modified SHA1 algorithm with a generated hash value of 1280 bit was being put to test using the crackstation tool. The generated 1280 bit hash size was looked up inside its databases which consists of existing familiar hash algorithms. This crackstation tool is being designed to search fo a possible match. Even though the tool acknowledged the 1280 bit hash size as input, unfortunately it could not produced the corresponding text value of the hash code stored in its database which produced a result of unrecognized hash format. It can be observed from the figure that the color code is red. This implied that the hash code is the unrecognized format and its type is unknown or cannot be found in its database. This is due to the fact that crackstation tool will work only on the current hashing algorithm such as sha256, MD160, MD5, sha512, etc.



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

```
5f4dcc3b5aa765dfff8327deb882cf998512135402a8e2c23365a3a9e1cba05b6c28
3baa07bd805c971ca782eb6185dd367e988e675c4456c171b8dc634f2196b9c29e8b
3f3c154ca8935dc04121922de20a2ba12d69c3ef4ac9fa9ad00e94f96d928b9e28c2
2d329b796dd1f57f712c280a52da16a58996bbd43998766011a5267c6b5f678ad337
45bf29df7f4314b237bf2fb66b72d51de460fb2228058745
```

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
5f4dcc3b5aa765dfff8327deb882cf998512135402a8e2c23365a3a9e1cba05b 6c283baa07bd805c971ca782eb6185dd367e988e675c4456c171b8dc634f2196 b9c29e8b3f3c154ca8935dc04121922de20a2ba12d69c3ef4ac9fa9ad00e94f9 6d928b9e28c22d329b796dd1f57f712c280a52da16a58996bbd43998766011a5 267c6b5f678ad33745bf29df7f4314b237bf2fb66b72d51de460fb2228058745	Unknown	Not found.

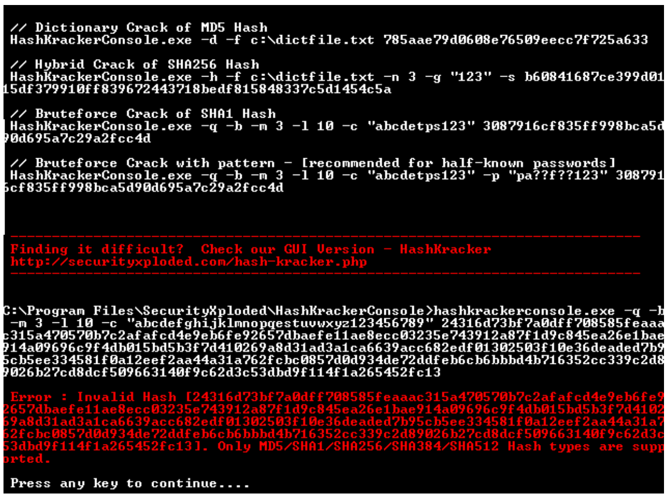
Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Figure 2. Cracking the Modified SHA1 using CrackStation

4.1 Testing a brute force attack

Other common attack in a hash algorithm is brute force attack. A brute-force attack attempts each potential permutation of characters up to a known length. These kinds of attacks are very arithmetically costly, and are typically the least effective in terms of cracking hashes per processor time, but passwords are ultimately cracked. In creating a password, it is advised to be long enough with the combination of both letters and numbers so that searching all possible character string would be difficult. The researchers used the hashkracker console program to perform this attack on the hash code of "MYPASS" generated by the modified SHA1 to measure its reliability. HashKracker Console is the all-in-one command-line tool to find out the password from the Hash. Currently, it supports password recovery from following popular Hash types such as MD5, SHA1, SHA256, SHA384, and SHA512.

Shown in Figure 3, the attack was not successful in the modified SHA1 since the tool was not able to identify the type of the hash code the modified SHA1 generated. Moreover, the hacker should know the source code of the modified SHA1 in order for him to carry out a brute force attack.

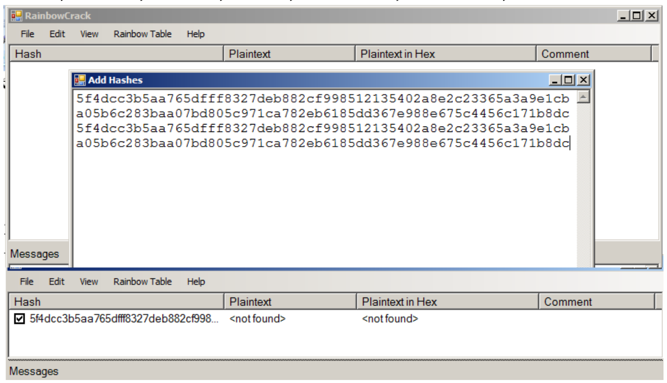


```
// Dictionary Crack of MD5 Hash
HashKrackerConsole.exe -d -f c:\dictfile.txt 785aae79d0608e76509eecc7f725a633

// Hybrid Crack of SHA256 Hash
HashKrackerConsole.exe -h -f c:\dictfile.txt -n 3 -g "123" -s b60841687ce399d01
15df379910ff839672443718bedf815848337c5d1454c5a

// Bruteforce Crack of SHA1 Hash
HashKrackerConsole.exe -q -b -m 3 -l 10 -c "abcdetps123" 3087916cf835ff998bca5d
90d695a7c29a2fcc4d

// Bruteforce Crack with pattern - [recommended for half-known passwords]
HashKrackerConsole.exe -q -b -m 3 -l 10 -c "abcdetps123" -p "pa??f??123" 308791
6cf835ff998bca5d90d695a7c29a2fcc4d


--------------------------------------------------------------------------------
Finding it difficult?  Check our GUI Version - HashKracker
http://securityxploded.com/hash-kracker.php
--------------------------------------------------------------------------------


C:\Program Files\SecurityXploded\HashKrackerConsole>hashkrackerconsole.exe -q -b
 -m 3 -l 10 -c "abcdefghijklmnopqestuvwxyz123456789" 24316d73bf7a0dff708585feaaa
c315a470570b7c2afafcd4e9eb6fe92657dbaefe11ae8ecc03235e743912a87f1d9c845ea26e1bae
914a09696c9f4db015bd5b3f7d410269a8d31ad3a1ca6639acc682edf01302503f10e36deaded7b9
5cb5ee334581f0a12eef2aa44a31a762fcbc0857d0d934de72ddfeb6cb6bbbd4b716352cc339c2d8
9026b27cd8dcf509663140f9c62d3c53dbd9f114f1a265452fc13

Error : Invalid Hash [24316d73bf7a0dff708585feaaac315a470570b7c2afafcd4e9eb6fe9
2657dbaefe11ae8ecc03235e743912a87f1d9c845ea26e1bae914a09696c9f4db015bd5b3f7d4102
69a8d31ad3a1ca6639acc682edf01302503f10e36deaded7b95cb5ee334581f0a12eef2aa44a31a7
62fcbc0857d0d934de72ddfeb6cb6bbbd4b716352cc339c2d89026b27cd8dcf509663140f9c62d3c
53dbd9f114f1a265452fc13]. Only MD5/SHA1/SHA256/SHA384/SHA512 Hash types are supp
orted.

Press any key to continue....
```

Figure 3. Using HashKracker Console to perform brute force attack on the Modified SHA1

4.2 Testing a Rainbow Table Attack

Rainbow tables are considered as a time-memory exchange technique. They are similar to lookup tables, although they sacrifice the speed to form the lookup tables smaller. Furthermore, a Rainbow Crack table is a tool that was developed by attacker to access the password by simply examining only at a hashed value. Figure 4 showed a rainbow table attack using RainbowCrack software. Results showed that the attack was unsuccessful to crack the hash value of "PASSWORD" which is 1280-bit generated from the modified SHA1 and the result was "not found" in its database. This is due to the fact that the database only consists of hash values of the current hash algorithms such as MD4, SHA1, SHA2, MD5, SHA256, SHA512, etc.


RainbowCrack
File Edit View Rainbow Table Help
Add Hashes
Messages
File Edit View Rainbow Table Help
Messages

```
5f4dcc3b5aa765dfff8327deb882cf998512135402a8e2c23365a3a9e1cb
a05b6c283baa07bd805c971ca782eb6185dd367e988e675c4456c171b8dc
5f4dcc3b5aa765dfff8327deb882cf998512135402a8e2c23365a3a9e1cb
a05b6c283baa07bd805c971ca782eb6185dd367e988e675c4456c171b8dc
```

Hash	Plaintext	Plaintext in Hex	Comment
☑ 5f4dcc3b5aa765dfff8327deb882cf998...	<not found>	<not found>	

Figure 4. Testing for rainbow table attack using Rainbow Crack

4.3 Securing login authentication in a client server communication using the modified SHA1

Shown in Figure 5 and Figure 6 is how to secure the login authentication of a client-server communication using the modified SHA1 through login authentication. Users are advised to create a password that is not easy to guess. Thus in the login form, the only password with a combination of characters and numbers with a capital letter will be accepted. These will prevent the attackers from guessing the right password. Then the script found in the server will generate a hash value of the message or password inputted by the user into 1280 bit and then compares it to the hash value of the password previously stored in the database during registration. If the hash code of the password entered by the user matches with the hash code stored in the database, then the user is able to access the system. However, if the hash value does not match, then the server will not grant the request from the user.

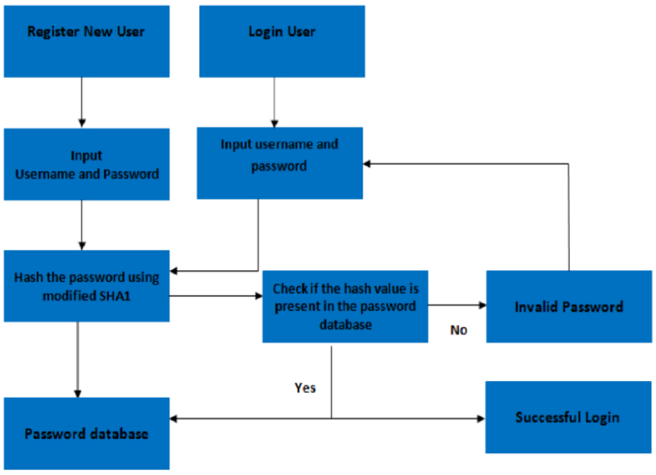
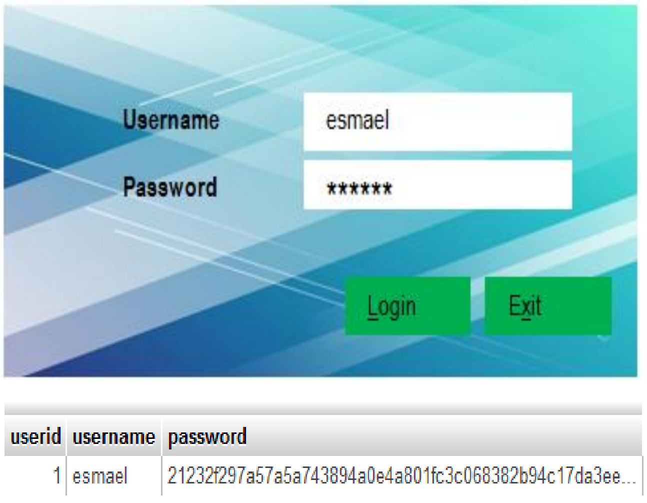

Register New User
Login User
Input Username and Password
Input username and password
Hash the password using modified SHA1
Check if the hash value is present in the password database
No
Invalid Password
Yes
Password database
Successful Login

Figure 5. Securing a client-server communication using modified SHA1



Username esmael
Password ******

userid	username	password
1	esmael	21232f297a57a5a743894a0e4a801fc3c068382b94c17da3ee...

Figure 6. Login Form with stored 1280-bit Modified SHA1 hash value stored in the server's password database.

5. Conclusion

In this paper, the modified SHA1 has been proposed and developed to expand its hash size up to 1280 bit to secure web applications through its login authentication against brute force, online cracking tools and rainbow table attacks. The expansion was done by allocating 32 buffer registers for variables A, B, C and D at 5 bytes and generating 4 buffer registers in every round inside the compression function of the original SHA1 for 8 times. During testing, it showed that the hash code of the modified SHA1 was not cracked using powerful online cracking tools, brute force and rainbow table attacks that are available online.

References

[1] G. Raj, R. Kesireddi, and S. Gupta, "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), 2015.

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

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

[4] L. Zhong, W. Wan and D. Kong, "JAVAWEB LOGIN AUTHENTICATION BASED ON IMPROVED MD5 ALGORITHM," International Conference on Audio, Language and Image Processing (ICALIP), 2016.

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

[6] P. Walia and V. Thapar, "Implementation of New Modified MD5-512 bit Algorithm for Cryptography," International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 1, no. 6, 2015.

[7] S. Sonh, "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, vol. 16, no. 5, pp. 1015-1022, 2012.

[8] V. Kapoor, "A new security system using ECC AND MD5," International Journal of Engineering Research & Technology (IJERT), 2015.

[9] M. Stevens, P. Karpman, and T. Peyrin, "Freestart Collision for Full SHA-1," Annual International Conference on the Theory and Applications of Cryptographic Techniques, vol. 9665, pp. 459-483, 2016.

[10] A. Bhandari, "Enhancement of MD5 Algorithm for Secured Web Development," Journal of Software, vol. 12, no.4, pp. 240-252, 2017.

[11] Y. Sasaki, "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, no.1, pp. 26-38, 2015.

[12] A. Kasgar, J. Agrawal and S. Sahu, "New Modified 256-bit MD5 Algorithm with SHA Compression Function," International Journal of Computer Applications, vol. 42, no. 12, 2012.

[13] A. Asbduvaliyev, S. Lee, and Y.K. Lee, "Modified SHA1 hash function (mSHA1)," Journal of Computing and Information Technology, vol. 21, no.4, 2016.

[14] R. Rivest, "The MD4 message digest algorithm," Conference on the Theory and Application of Cryptography, pp. 303–311, 1990.

[15] T. Lakshmanan, and M. Muthusamy, "A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes," The International Arab Journal of Information Technology, vol.9, no.3, pp. 262 – 267, 2012.

[16] M. Hassouna, B. Barry, and E. Bashier, "A Short Certificateless Digital Signature Scheme," The Proceedings of the International Conference on Digital Information Processing, Data Mining, and Wireless Communications, Dubai, UAE, 2015.

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

[18] M. Alam, and S. Ray, "Design of an Intelligent SHA-1 Based Cryptographic System: A CPSO Based Approach," International Journal of Network Security, vol. 15, no.6, pp.465-470, 2013.

[19] A. Kasgar, J. Agrawal and S. Sahu, "New Modified 256-bit MD5 Algorithm with SHA Compression Function," International Journal of Computer Applications, vol. 42, no. 12, 2012.

[20] C. San Jose, B. Gerardo, and B. Tanguilig, "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, 2015.

[21] R. Siddhartha, "Advanced SHA-1 Algorithm Ensuring Stronger Data Integrity," International Journal of Computer Applications, vol. 130, no. 8, 2015.

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

[23] M.R. Baharon, Q. Shi, M. F. Abdollah, S.M. W. Mohamed, S.M.M Yassin, and A. Idris, "An Improved Fully Homomorphic Encryption Scheme for Cloud Computing," International Journal of Communication Networks and Information Security (IJCNIS), vol. 10, no. 3, 2018.

[24] H. Tiware, and K. Asawa, "A Secure Hash Function MD-192 With Modified Message Expansion," International Journal of Computer Science and Information Security, (IJCSIS),vol. 7, no.2, 2010.

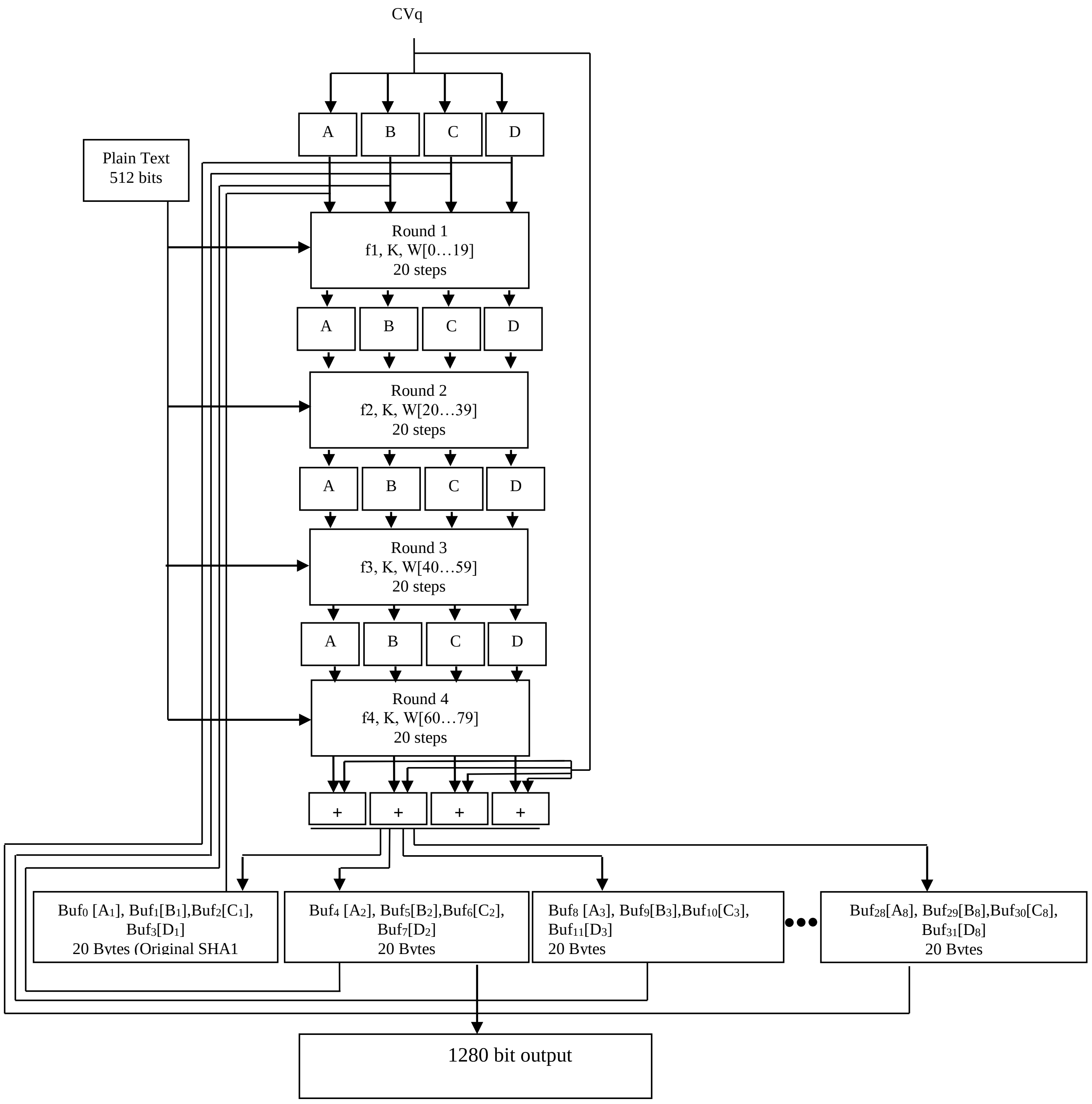

CVq
A
B
C
D
Plain Text 512 bits
Round 1 f1, K, W[0…19] 20 steps
Round 2 f2, K, W[20…39] 20 steps
Round 3 f3, K, W[40…59] 20 steps
Round 4 f4, K, W[60…79] 20 steps
+
Buf0 [A1], Buf1[B1],Buf2[C1], Buf3[D1] 20 Bytes (Original SHA1
Buf4 [A2], Buf5[B2],Buf6[C2], Buf7[D2] 20 Bytes
Buf8 [A3], Buf9[B3],Buf10[C3], Buf11[D3] 20 Bytes
Buf28[A8], Buf29[B8],Buf30[C8], Buf31[D8] 20 Bytes
1280 bit output

Figure 1. System Architecture of the Modified SHA1