IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 2, Ser. II (Mar - Apr 2020), PP 47-51
www.iosrjournals.org

Improving Security Through Hashing

Vishal Yadav[1], Harsh Agrawal[2], Asim Kazi[3],Yash Shah[4]
(Department of Information Technology, Vidyalankar Institute of Technology, Mumbai, India)

Abstract: *In today's scenario we totally depend on the internet for our everyday tasks. The amount of data transmitted through the internet is growing every day. The need of a hashing algorithm that can guarantee that the integrity and confidentiality of the data is maintained. The aim of the research project is to ensure secure transmission and reception of data with speed and reliability. This model uses the Secure Hash Algorithm (SHA) that creates and appends hashes of the data during transmission. This project aims to discuss the drawbacks and shortcomings of the now used hashing algorithms and come up with enhanced security features along with maintaining speed.*

Key Word*:Hashing, Data Integrity,Cryptography,SHA.*

Date of Submission: 19-04-2020 Date of Acceptance: 03-05-2020

I. Introduction

In this era of digitization, the foremost requirement is information security which is normally secured by some authentication process. Authentication is one of the most important requirements to secure information. There exist various methods for authentication (e.g. passwords, PINs etc). Password based systems are easier to implement than other existing methods and most generally used method for authentication. But to store user passwords within databases as plaintext is a major blunder made by developers. Password security is a major issue for any authenticating process and different researches in past have proposed different techniques like hashing, salting, honeywords to make the process most secured. And still we see security breaches everyday in the systems using traditional security algorithms. The aim of the research project is to ensure secure transmission and reception of data with speed and reliability. This model uses the Secure Hash Algorithm (SHA) that creates and appends hashes of the data during transmission. This project aims to discuss the drawbacks and shortcomings of the now used hashing algorithms and come up with enhanced security features along with maintaining decent speed.

II. Security Paradigm

Some of the properties of a good hashing algorithm are:

1.Computationally Efficient

First and foremost, hash functions must be computationally efficient.Computer must be able to perform a hash function's mathematical labor in an extremely short period of time.This property is probably somewhat obvious. If an ordinary computer needed several minutes to process a cryptographic hash function and receive the output, it would not be very practical. To be useful, hash functions must be computationally efficient.In reality, this is not as large of a concern as it was 40 or 50 years ago. Nowadays, an average home computer can process an advanced hash function in just a small fraction of a second.

2. Deterministic

Cryptographic hash functions must be deterministic. In other words, for any given input, a hash function must always give the same result.If a cryptographic hash function were to produce different outputs each time the same input was entered, the hash function would be random and therefore useless. It would be impossible to verify a specific input, which is the whole point of hash functions— to be able to verify that a private digital signature is authentic without ever having access to the private key.

3. Pre-Image Resistant

The output of a cryptographic hash function must not reveal any information about the input. This is called pre-image resistance.It's important to note that cryptographic hashing algorithms can receive any kind of input. The input can be numbers, letters, words, or punctuation marks. It can be a single character, a sentence from a book, a page from a book, or an entire book.However, a hash function will always produce a fixed-length output. Regardless of what the input is, the output will be an alphanumeric code of fixed length.

4. Collision Resistant

The final property that all cryptographic hash functions must have is what's known as collision resistance. This means that it must be extremely unlikely— in other words, practically impossible— to find two different inputs that produce the same output.The inputs to a hash function can be of any length. This means there are infinite possible inputs that can be entered into a hash function.However, outputs are of a fixed length. This means that there are a finite number— albeit an extremely large number— of outputs that a hash function can produce. A fixed-length means a fixed number of possibilities.Since the number of inputs are essentially infinite, but the outputs are limited to a specific number, it is a mathematical certainty that more than one input will produce the same output.The goal is to make finding two inputs that produce the same output so astronomically improbable that the possibility can be practically dismissed outright. It should not pose a risk.

III. Literature Survey

A several analysis by numerous researches is work on SHA and its variants. The outline of the analysis is reviewed as follows:

3.1 Performance Analysis of SHA Algorithms (SHA-1 and SHA-192): A Review

[1]There are many secure hash algorithms are available here. All these algorithms are iterative, one-way hash functions to produce a message that can process for condensed representation called a message digest. The existing algorithms enable the message's integrity for messages: there is high probability that any change to the message, results in a different message digest. For the authentication codes and verification of digital signatures his property is very useful, and also in the random numbers (bits) generation. The existing algorithms differ mostly in the number of bits of security that are provided for the information being hashed this is directly related to the message digest length. When an existing secured hash algorithm is used in conjunction with other algorithm, there may be requirements specified elsewhere that require the use of a existing secured hash algorithm with a certain number of bits of security. This paper presenting combined study of SHA-160 and SHA-192 algorithm. Experimental results are presenting overall observation of these two algorithms.

3.2 Review Paper on Secure Hashing Algorithm and Its Variants

[2]Linux is one the most widely used operating systems. With its inherent ubiquity come its many flaws. In this implementation the aspect of system security is considered. We have implemented a kernel patch which isolates the execution of ELF files in Ubuntu. The signatures of these files are verified before the loading and execution can proceed. We verify the path of the file and its hash value. Any change in path or the contents of the file and hence change in its hash value will prevent it from being executed and hence a safe execution environment is provided. The kernel with the security patch takes an average of 0.006 seconds more time than the kernel without the patch which means the user of such a system will not feel any delays in execution.

3.3 A Comparative Analysis of SHA and MD5 Algorithm

[3]This paper is based on the performance analysis of message digest 5 and secure hashing algorithm. These two topics are related with cryptography and cryptography is an extension of cryptology and cryptanalysis. The purpose of this paper is that to compare the time taken to build a hash as well as it also compares the bit rate passes through a hash value. Here we are going to perform a deep analysis for these two algorithms.

3.4 Secure and Efficient Integrity Algorithm based on Existing SHA Algorithms

[4]In today's world every person relies on internet for various purposes. There is always a need to take appropriate measures for getting secure communication all the way throughout this unsecure internet. Integrity is one of the most significant factors in the communication scenario. There are various algorithms that ensure the integrity but almost all are either not secure or not efficient. This paper highlights some of such algorithms and also introduces an integrity algorithm and also proves its efficiency with its implementation result.

3.5 A Comparative Study on Different Hashing Algorithms

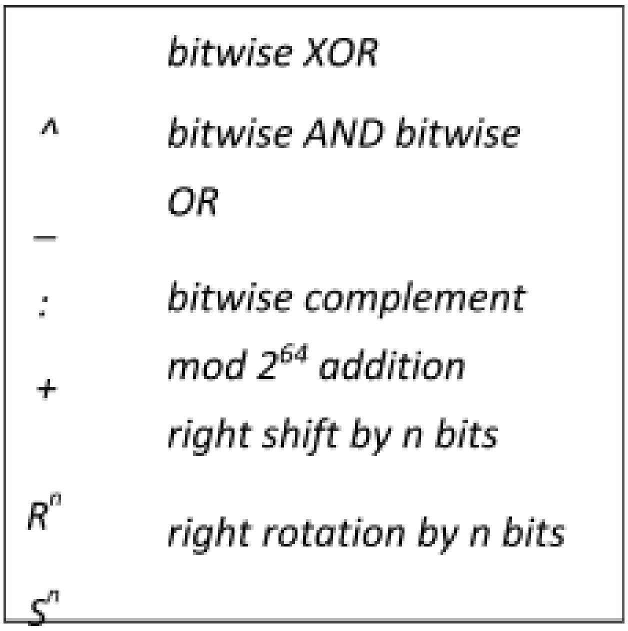
[5] Passwords play an important role in daily life in various computing applications and play a critical role in online authentication. The main aim for using passwords is to restrict unauthorized users to access the system. Passwords are necessary to provide the security to the users because of many flaws in the conventional password systems. Unfortunately, passwords suffer from two intractable problems: password cracking and password theft. So we use Password hashing to protect password. Password hashing technique allows users to remember simple passwords and have them hashed to create secure passwords. This paper describes widely used hash algorithms and comparative analysis of different hash algorithms which are used in password hashing

for making awareness of attacks and selection of hashing method in a particular scenario analysis by numerous researches is work on SHA and its variants.

IV. Methodology

SHA-640 is a variant of SHA-512 which operates on ten 64 bit buffers. All the other computation remains the same

SHA-512 is a variant of SHA-256 which operates on eight 64-bit buffers. The message to be hashed is rst (1) padded with its length in such away that the result is a multiple of 1024 bits long, and then (2) parsed into 1024-bit message blocks M(1) ; M(2) ;:::;M(N) . The message blocks are processed one at a time: Beginning with a xed initial hash value H(0) , sequentially compute H(i) = H(i1) + CM(i) (H(i1)); where C is the SHA-512 compression function and + means word-wise mod 512 addition. H(N) is the hash of M



	bitwise XOR
^	*bitwise AND bitwise*
	OR
—	
:	*bitwise complement*
	mod 2^{64} addition
+	
	right shift by n bits
R^n	*right rotation by n bits*
S^n	

All of these operators act on 64-bit words. The initial hash value H(0) is the following sequence of 64-bit words (which are obtained by taking the fractional parts of the square roots of the first eight primes): H(0) 1 = 6a09e667f3bcc908 H(0) 2 =bb67ae8584caa73b H(0) 3 =3c6ef372fe94f82b
H(0) 4 =3c6ef372fe94f82b H(0) 5 = 510e527fade682d1 H(0) 6 = 9b05688c2b3e6c1f H(0) 7 =1f83d9abfb41bd6b H(0) 8 =5be0cd19137e2179.
But for SHA-640 we take two more 64-bit words.The next two words are obtained by taking the fractional parts of square roots of the ninth and tenth primes H(0)9=cbbb9d5dc1059ed8 H(0)10=629a292a367cd507

3)Preprocessing Computation of the hash of a message begins by preparing the message:

1.Pad the message in the usual way: Suppose the length of the message M, in bits, is `. Append the bit \1" to the end of the message, and then k zero bits, such that the length of the message is 128< than multiple of 1024. For eg. for a 1919 bit no. padding bits=(1919-128)mod 1024 .To this append the 128-bit block which is equal to the number written in binary. : The length of the padded message should now be a multiple of 1024 bits.

2. Parse the message into N 1024-bit blocks M(1); M(2) ;:::;M(N) . The first 64 bits of message block i are denoted M(i) 0 , the next 64 bits are M(i) 1 , and so on up to M(i) 15 . We use the big-endian convention throughout, so within each 64-bit word, the leftmost bit is stored in the most significant bit position. Main loop The hash computation proceeds as follows: For i = 1 to N (N = number of blocks in the padded message) f Initialize registers a; b; c; d; e; f ; g; h. And for SHA-640 we initialize two mre registers i ; and j. with the (i 1)st intermediate hash value (= the initial hash value when i = 1) a H(i1) 1 b H(i1) 2 . . . h H(i1) 10
Apply the SHA-512 compression function to update registers a; b;c;d;e;f;g;h;I;j For p= 0 to 63 f Compute Ch(e; f ; g), Maj(a; b; c), and Wp (see Denitions below)
T1= = h + S1 + ch(e;f;g) + k[i] + w[i]

T2= S0 + maj(a;b;c) + f
j = i+temp2
i = h+ temp1
h = g
g = f
f = e
e = d + temp1
d = c
c = b
b = a
a =i+ (temp1 + temp2)
Compute the ith intermediate hash value H(i)

Denitions: Six logical functions are used in SHA-512. Each of these functions operates on 64-bit words and produces a 64-bit word as output. Each function is defined as follows: Ch(x; y; z)=(x ^ y) (:x ^ z) M aj(x; y; z)=(x ^ y) (x ^ z) (y ^ z) S0 = rightrotate(a, 28) ^ right rotate(a, 34) ^ right rotate(a, 39)
S1 = rightrotate(e, 14) ^ right rotate(e, 18) ^ right rotate(e, 41)

Expanded message blocks W0; W1;:::;W63 are computed as follows via the SHA-512 message schedule: Wp = M(i) p for p = 0; 1;:::; 15, and

```
For p = 16 to 63
        s0 = rightrotate(w[p - 15], 1) ^ rightrotate(w[p - 15], 8) ^ (w[p - 15] ,7)
        s1 = rightrotate(w[i - 2], 19) ^ rightrotate(w[p - 2], 61) ^ (w[p- 2] , 6)
        w[i] = (w[p - 16] + s0 + w[p - 7] + s1)
```

Denitions, continued A sequence of constant words, K0;:::;K63; is used in SHA-512. In hex, these are given by

428a2f98d728ae22, 7137449123ef65cd, b5c0fbcfec4d3b2f, e9b5dba58189dbbc, 3956c25bf348b538, 59f111f1b605d019, 923f82a4af194f9b, ab1c5ed5da6d8118, d807aa98a3030242, 12835b0145706fbe, 243185be4ee4b28c, 550c7dc3d5ffb4e2, 72be5d74f27b896f, 80deb1fe3b1696b1, 9bdc06a725c71235, c19bf174cf692694, e49b69c19ef14ad2, efbe4786384f25e3, 0fc19dc68b8cd5b5, 240ca1cc77ac9c65, 2de92c6f592b0275, 4a7484aa6ea6e483, 5cb0a9dcbd41fbd4, 76f988da831153b5, 983e5152ee66dfab, a831c66d2db43210, b00327c898fb213f, bf597fc7beef0ee4, c6e00bf33da88fc2, d5a79147930aa725, 06ca6351e003826f, 142929670a0e6e70, 27b70a8546d22ffc, 2e1b21385c26c926, 4d2c6dfc5ac42aed, 53380d139d95b3df, 650a73548baf63de, 766a0abb3c77b2a8, 81c2c92e47edaee6, 92722c851482353b, a2bfe8a14cf10364, a81a664bbc423001, c24b8b70d0f89791, c76c51a30654be30, d192e819d6ef5218, d69906245565a910, f40e35855771202a, 106aa07032bbd1b8, 19a4c116b8d2d0c8, 1e376c085141ab53,
2748774cdf8eeb99, 34b0bcb5e19b48a8, 391c0cb3c5c95a63, 4ed8aa4ae3418acb, 5b9cca4f7763e373, 682e6ff3d6b2b8a3, 748f82ee5defb2fc, 78a5636f43172f60, 84c87814a1f0ab72, 8cc702081a6439ec, 90befffa23631e28, a4506cebde82bde9, bef9a3f7b2c67915, c67178f2e372532b, ca273eceea26619c, d186b8c721c0c207, eada7dd6cde0eb1e, f57d4f7fee6ed178, 06f067aa72176fba, 0a637dc5a2c898a6, 113f9804bef90dae, 1b710b35131c471b, 28db77f523047d84, 32caab7b40c72493, 3c9ebe0a15c9bebc, 431d67c49c100d4c, 4cc5d4becb3e42b6, 597f299cfc657e2a, 5fcb6fab3ad6faec, 6c44198c4a475817)

These are the first sixty four bits of the fractional parts of the cube roots of the first sixty-four prime numbers)

V. Results

Traditional(SHA 512)	Modified(SHA-640)
0.50976	0.50195
0.47070	0.50195
0.45703	0.48828
Average: 0.47916	Average: 0.48424

Avalanche effect calculated in three different cases.

- Time Complexity

Hash	#1 (ms)	#2 (ms)	#3 (ms)	Average(ms)
	1073	1055	1050	
SHA-512				1059.3
	1204	1197	1195	
Modified Algorithm				1198.6

The algorithm is analyzed on the following parameters:

- Avalanche Analysis: Avalanche effect was calculated for three cases. In two out of those 3 cases, the avalanche effect of modified algorithm was found to be greater than the traditional SHA512 algorithm.

The average avalanche effect of modified algorithm gave a value of 0.48424 which was greater than 0.47916 the average value given by traditional SHA 512.

- Time Complexity: The average time complexity of modified algorithm was found to be greater than the average time complexity of traditional SHA 512.

VI. Conclusion

Hashing is an important concept in the security domain which satisfies various requirements.We've studied the SHA-512 algorithm throughly and tried to improvise it in our new version SHA-640. The modified version of SHA-640 performs better in some cases as you can see in the results above.

References

[1]. Piyush Garg, and Namita Tiwari ,Performance Analysis of SHA Algorithms (SHA-1 and SHA-192): A Review,International Journal of Computer Technology and Electronics Engineering (IJCTEE)Volume 2, Issue 3, June 2012

[2]. Priyanka Vadher a and Bhumika Lall,Review Paper on Secure Hashing Algorithm and Its Variants,International Journal of science and reasearh (IJSR)

[3]. Piyush Gupta, and Sandeep Kumar,A Comparative Analysis of SHA and MD5 Algorithm,(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 4492-4495

[4]. Snigdha Soni and Pratap Singh ,Secure and Efficient Integrity algorithm based on Existing SHA Algorithms,International Journal of Computer Applications ,Volume 113 - Number 11,2015

[5]. C.G Thoma s and Robin Thoma s Jose ,A Comparative Study on Different Hashing Algorithms ,International Journal of Innovative Research in Computerand Communication Engineering,Vol. 3, Special Issue 7, October2015