Secure and Efficient Integrity Algorithm based on Existing SHA Algorithms

Snigdha Soni
Department of Computer Science & Engineering,
Oriental Institute of Science and Technology
Bhopal, India

Sandeep Pratap Singh
Department of Computer Science & Engineering,
Oriental Institute of Science and Technology
Bhopal, India

ABSTRACT
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.

Keywords
Computer Security, SHA, Hash, Message Digest

1. INTRODUCTION
In present technology, security is always an important concern. Mainly security is based on three parameters i.e. integrity, confidentiality and authenticity. This paper focuses on integrity. Many cryptographic hash functions are designed in order to provide integrity over data. Hash function generates a message digest of fixed length. Hash function is designed in such a way that it contains the following properties:

a. It should be easy to compute the hash value for any given message.

b. It should be infeasible to generate a message that has a given hash.

c. It should be infeasible to modify a message without changing the hash.

d. It should be infeasible to find two different messages with the same hash.[1, 10]

There are many cryptographic hash algorithms: MD4, MD5, SHA-0, SHA-1 and SHA-2, but all these algorithms are either not secure or not efficient in terms of time. Hash algorithm uses a message and generates a message digest of fixed length. This message digest is appended with the message and then sent to the other end. At the other end, message and digest are separated and message again passes through the hash function and generates another digest. If both the digest are same it means message is original else the message is discarded. The key property of a hash algorithm is that it is impossible to generate a message from digest.

MD-4 is a cryptographic hash algorithm designed by a Ronald Rivest in the year of 1990. It generates a message digest of 128 bits. MD-4 is not much secure, first full attack on MD-4 is found in the year of 1995. Latest a 2007 attack found a collision on MD-4 in less than 2 hash operations [11].

MD-5 is another cryptographic hash algorithm that generates a message digest of 128 bits. It was developed by Ron Rivest in 1996. Again a 2013 attack by Xie Tao, Fanbao Liu, and Dengguo Feng breaks MD5 collision resistance in 2^{18} times. This attack runs in less than a second on a regular computer [11].

SHA-0 is the first algorithm proposed in SHA family. It generates a message digest of 160 bits. An attack in 2008 applying the boomerang attack brought the complexity of finding collisions down to $2^{33.6}$, which is estimated to take 1 hour on an average PC [11].

SHA-1 is the most popular hash algorithm among all the existing algorithms, the reason of its popularity is its efficiency. It generates a message digest of 160 bits but a 2011 attack by Marc Stevens produces hash collisions with a complexity between $2^{60.3}$ and $2^{65.3}$ operations [11].

SHA-2 is the most secure algorithm among all the existing algorithms but still it doesn't get that much popularity because of its inefficiency.

After that many researchers have proposed their own algorithms because all the existing algorithms are either breakable or not efficient.

In [2], Authors have proposed there algorithm which generates a message digest of 192 bits. The internal structure of SHA-192 is almost similar to the other SHA algorithms.

In [1], Authors have merged the SHA-192 and MD-5 and proposed their own algorithm. This algorithm also generates a digest of 192 bits.

In this paper, it proposed its own algorithm in such a way that it is not only time efficient but also highly secure. Also proposed algorithm is compared with the existing research work of SHA-192 and SHA-1.

2. PROPOSED ALGORITHM
Internal design of proposed algorithm is different from the existing SHA algorithm. Proposed algorithm uses 10 chaining variable of 16 bits while SHA-1 uses 5 chaining variable of 32 bits and SHA-192 uses 6 chaining variable of 32 bits. Proposed algorithm generates a message digest of 160 bits and takes 64 rounds for each 512 bits chunk. The modified structure of proposed SHA-160 algorithm is given in Fig 1.

Steps of Proposed Algorithm are as follows

a. Padding: The first step in Proposed SHA is to add padding bits to the original message. The aim of this

step is to make the length of the original message equal to a value, which is 64 bits less than an exact multiple of 512. We pad message M with one bit equal to 1, followed by a variable number of zero bits.

b. Append length: After padding bits are added, length of the original message is calculated and expressed as 64 bit value and 64 bits are appended to the end of the resultant message of step 1.

c. Divide the input into 512 bit blocks: Divide the input message into blocks, each of length 512 bits, i.e. cut M into sequence of 512 bit blocks M^1, M2…..M^N. Each of M^i parsed into thirty-two 16 bits words M^i_0, M^i_1……...M^i_{32}

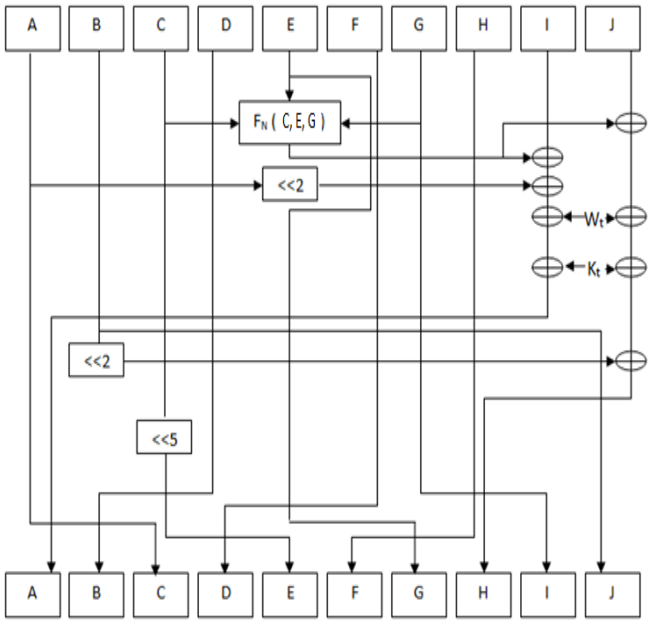

Fig 1: Elementary Function of Proposed SHA

d. Initialize chaining variables the hash is 160 bits used to hold the intermediate and final results. Hash can be represented as ten 16 bits word registers, A,B,C,D,E,F,G,H,I,J,K. Initial values of these chaining variables are:

A = 6745

B = 2301

C = EFCD

D = AB89

E = 98BA

F = DCFE

G=1032

H=5476

I=C3D2

J=E1F0

The compression function maps 160 bits value H=(A,B,C,D,E,F,G,H,I,J) and 512 bit block M^i into 160 bits value. The shifting of some of the chaining variables by 10 bits in each round will increase the randomness in bit change in the next successive routines. If the minimum distance of the similar words in the sequence is raised then the randomness significantly raises. A different message expansion is employed in this hash function in such a way that the minimum distance between the similar words is greater compared with existing hash functions.

e. Processing: After pre-processing is completed each message block is processed in order using following steps:

I) for i = 1 to N prepare the message schedule.

$M^i t$, $0 \le t \le 31$

$Wt = (W_{t-6}$ XOR W_{t-16} XOR W_{t-14} XOR $W_{t-32}) <<1 \quad 32 \le t \le 63$

II) Initialize the ten working variables A,B,C,D,E,F,G,H,I,J with $(i-1)^{st}$ hash value.

III) for t = 0 to 63

{

Temp = (A<<2) XOR F1 (C, E, G) XOR I ^K_t ^ W_t

I = G;

G = E;

E = C<<5

C = A;

A = Temp

Temp = B<<2 XOR F1 (C, E, G) XOR J XOR K_t XOR W_t

e2 = a2;

a2 = b2;

b2 = c2;

c2 = d2;

d2 = step5;

}

Where Kt is a constant defined by a Table 4.1, F1 is a bitwise Boolean function, for different rounds defined by,

F1 (C, E, G) = IF C THEN E ELSE G

F1 (C, E, G) = C XOR E XOR G

F1 (C, E, G) = MAJORITY (C, E, G)

F1 (C, E, G) = C XOR E XOR G

Where the "IF….THEN……ELSE "function is defined by

IF C THEN E ELSE G = (CΛE) V ((¬C) ΛG)

And "MAJORITY" function is defined by

MAJORITY (C, E, G) = (CΛE) V (EΛG) V (GΛC)

Also, ROTL is the bit wise rotation to the left by a number of positions specified as a superscript.

IV) $H0^{(i)} = A + H0^{(i-1)}$

$H1^{(i)} = B + H1^{(i-1)}$

$H2^{(i)} = C + H2^{(i-1)}$

$H3^{(i)} = D + H3^{(i-1)}$

$H4^{(i)} = E + H4^{(i-1)}$

$H5^{(i)} = F + H5^{(i-1)}$

$H6^{(i)} = G + H6^{(i-1)}$

$H7^{(i)} = H + H7^{(i-1)}$

$H8^{(i)} = I + H8^{(i-1)}$

$H9^{(i)} = J + H9^{(i-1)}$

Table 1: Coefficients of each round

Rounds	**Steps**	**F1**	**Kt**
1	0-15	IF	FA92
2	16-31	XOR	6ED9
3	32-47	MAJ	8F1B
4	48-64	XOR	CA62

3. PERFORMANCE ANALYSIS

This section provides a comparative analysis between existing and proposed hash algorithms. To evaluate the efficiency and the strength of hash algorithms mostly two parameters are considered. First, Time Efficiency to calculate its efficiency against time and avalanche effect to calculate the internal strength of proposed algorithm.

3.1 Timing Analysis

Timing analysis is used to calculate the time efficiency of proposed algorithm against the existing algorithm. It is a time of generating message digest for any given text.

Table 2: Timing Comparison between Proposed Algorithm, SHA-1, and SHA-192 algorithms

File Size in KB	Algorithms (Time in Seconds)		
	SHA-160	Proposed SHA-160	SHA-192
5 KB	0.187	0.109	0.639
10 KB	0.483	0.374	1.138
20 KB	1.981	1.435	3.151

A graphical representation for the Table 2 is shown in Fig 2, with blue line for execution time of existing SHA and Proposed SHA-160 algorithms. According to the graph, there is a tendency that execution time for Proposed SHA-160 algorithm, increases with file size. But required time for the execution through Proposed SHA-160 is much smaller than execution time for compared SHA-192 and SHA-1.

3.2 Security Analysis

Another parameter that is used to calculate the internal strength of a hash algorithm is avalanche effect. According to avalanche effect two texts having a difference of a single bit should generate digests that are 50 percent different from each other. An algorithm that is close to this condition is considered better than other.

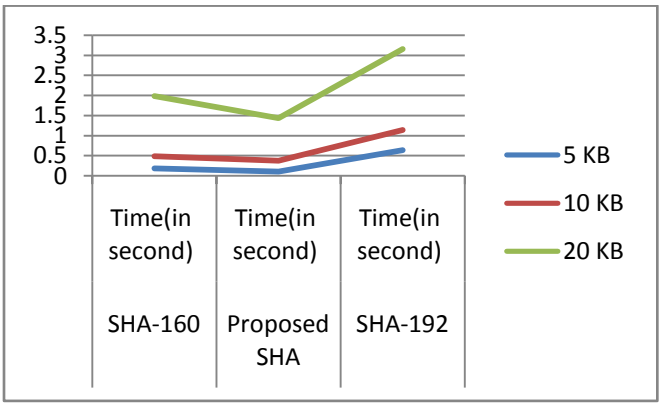

Fig 2: Timing Comparison between Proposed Algorithm, SHA-1, and SHA-192 algorithms

After calculating the avalanche effect, it is found that proposed algorithm is close to idle condition as compare to others.

Table 3: Avalanche effect of Proposed Algorithm, SHA-192, and SHA-1 algorithms

Algorithm	Avalanche Effect		
Message	**SHA-160**	**Proposed SHA-160**	**SHA-192**
	45.65 %	48.725 %	48.12 %

3.3 Space Analysis

Another parameter to evaluate the performance of proposed algorithm is space. Here, proposed algorithm generates a message digest of 160 bits therefore it requires a buffer that can hold 160 bits while on the other hand SHA-1 also requires a buffer to hold only 160 bits hence SHA-1 requires less space but on comparing it with SHA-192 algorithms that required a buffer to hold 192 bits, that requires more space, proposed algorithm comparatively requires less space and hence is more efficient in respect of space.

3.4 Analysis of Hash Code

If an algorithm generates a message digest of n bits than according to birthday attack it requires $2^{n/2}$ combinations to find the collision. SHA-1 generates 160 bits message digest,

hence it requires 2^{80} combinations to find the collision but in 2011 it is proven that collision can be found after 2^{61} combinations; SHA-192 generates 192 bits message digest hence it requires 2^{96} combinations and proposed algorithm generates 160 bits message digest therefore 2^{80} combinations are required to find collision. This value is enough large that supercomputer needs millions of year to solve this value.

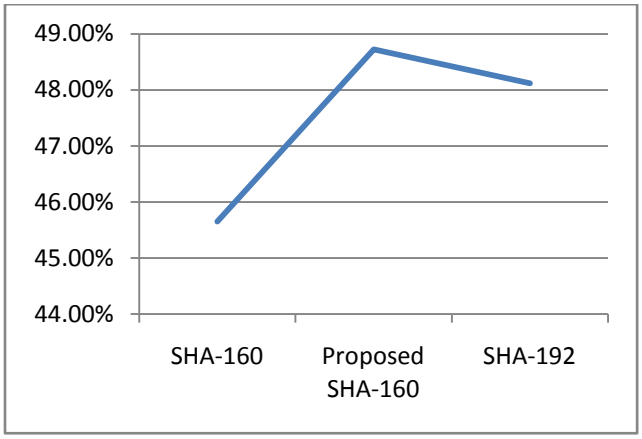

Fig 3: Avalanche effects of Proposed Algorithm, SHA-1 and SHA-192[1] algorithms

4. CONCLUSION

As discussed, there is always a demand of modification or replacement of existing algorithms with the modified or latest algorithms. This paper discussed one of the problems faced in integrity algorithms that all the existing algorithms are either proven breakable or not time efficient. This paper studied all such algorithms and also proposed its own integrity algorithm which is not only secure but also time efficient too. This paper shows its implementation results and also proved that proposed algorithm is the efficient and better option to be used in places where data integrity is considered essential. Authors have tested the above results on number of sample files and proposed there results.

Future Scope: There is always a chance of improvement; authors have tried their best to design this algorithm but still in future it can be further improved and be more efficient.

5. REFERENCES

[1] Garbita Gupta and Sanjay Sharma, *"Enhanced SHA-192 Algorithm with Larger Bit Difference"* IEEE International Conference on Communication Systems and Network Technologies, 2013

[2] L.Thulasimani and M.Madheswaran "Security and Robustness Enhancement of Existing Hash Algorithm" IEEE International Conference on Signal Processing Systems 2009.

[3] A new Hash Function Based on Combination of Existing Digest Algorithms pub 2007.

[4] The Collision Rate Tests of Two Known Message Digest Algorithms 2009.

[5] Harshvardhan Tiwari. A Secure Hash Function MD-192 with Modified Message Expansion" Vol. 7 No. 2 February 2010 International Journal of Computer Science and Information Security.

[6] Marc Stevens hash clash - Framework for MD5 & SHA-1 Differential Path Construction and Chosen-Prefix Collisions for MD5

[7] X. Wang, H. Yu and Y.L. Yin, "Efficient Collision Search Attacks on SHA-0",(Pub 2005)

[8] K. Matusiewicz and J. Pieprzyk, "Finding good differential patterns attacks on SHA-1", (Pub 2004), Available: http://eprint.iacr.org/2004/364.pdf

[9] William Stallings, "Cryptography and Network Security: Principles and Practice. Third edition, Prentice Hall.2003.

[10] Florent Chabaud, Antoine Joux, "Differential collisions in SHA-0," Advances in Cryptology-CRYPTO'98, LNCS 1462, Springer-Verlag, 1998.

[11] http://en.wikipedia.org/wiki/Secure_Hash_Algorithm

[12] Ricardo Chaves, Georgi Kuzmanov, Leonel Sousa, and Stamatis Vassiliadis " Cost-Efficient SHA Hardware Accelerators" IEEE transactions on very large scale integration (VLSI)Systems, VOL. 16, NO. 8, AUGUST 2008