Modified Secure Hashing algorithm(MSHA-512)

Ashoka SB[a], Lakshmikanth D[b]

[a]*Assistant professor , Maharani Cluster University*
[b]*Assistant professor , Maharani Cluster University*

ARTICLE INFO

Keywords:
SHA-512
Blockchain security
Rolled SHA-512
cyber security
MSHA-512

ABSTRACT

In recent year's security has become an important role in the field of Defense, Business, Medical and Industries.Different types of cryptography algorithms has been implemented in order to provide security with high performance. A hash function is a cryptography algorithm without a key such as MD5, SHA-family. Secure hash algorithm which is standardized by NIST as secured hashing in FIPS. In this paper we mainly focus on code optimization and increase the performance of SHA-512. To optimize and increase the performance of SHA-512 we have modified the algorithm and proposed Modified Secure hash algorithm(MSHA-512). It is a one-way hash function that can process a message to produce a condensed representation called a message digest.In this proposed algorithm within the limited rounds (40 rounds instead of 80 rounds) we can obtain exact result which is same as traditional SHA-512 algorithm. By using MSHA-512 algorithm we can reduce the time upto 50% for the same process, which increases the performance of the algorithm and helps to increase the flexibility of server in running streams.The MSHA-512 generates the same hash code as we generate in the SHA-512 for all different size of inputs. MSHA-512 is developed and designed to overcome the time complexity of SHA-512 and increase the performance of it by reducing the rounds.

1. Introduction

Internet and its technologies has made human life easier with respect to communication and sharing resources with higher speed. At the same time ,giving protection to an individual data will add more security towards internet technologies .cryptography is the most popular encryption system to protect information and to communicate with trusted sources by encrypting and decrypting the data. Modern cryptography deals with the confidentiality, Integrity and authentication.The cryptographic algorithms are basically classified into three categories: secret key cryptography, Public key Cryptography and Hash functions.This paper provides the detail study on the hash functions. Hash functions are one-way cryptography in which there is no key, since plain text is not recoverable from the cipher text. Message digest (MD5)[1] is one of the most popular one-way hash function used in authentication application purpose.MD5 produces a unique128 bit hash value for any input message. Hash collision often happens during generation of hash using hash functions.As discussed in the paper [2] MD5 hash function has been broken ,any determined hacker can produce two colliding assets in a matter of seconds. Paper contains study about probability of collisions happened in MD5 algorithm using birthday paradox. Later Secure Hash algorithm was introduced in the later years of 1993 where Secure Hash Algorithm for NIST's Secure Hash Standard (SHS)[3] is the most popular hash function implemented in digital signature algorithms, random number generators and keyed-hash message authentication codes.Where SHA-0 and SHA-1 produces a unique 160 bit hash value for input message.The paper[4] analyses the collision attack in SHA-1 family using identical-prefix collision attack.We have more chances of collision attack because of buffers with less size which tends to produce less output bits, to overcome this the buffer size is increased so the output will generate more bites with size having 256 ,384 or 512 bits and later it is called as SHA-2 Family.SHA-256 and SHA-512 are most widely used hash algorithms in block chain technology ,internet security like IPsec and digital certificates. SHA-512 is a hash function computed with 64 bit words. It produces 512 bits of message digest with 80 rounds of process where in each round hash function will be applied and hash is generated for next round. Hardware implementation of SHA-512 was discussed in paper [5].The performance of the SHA-512 depends on computing hash occurring in each round. The paper "SHA-512/256"[6] analyses the performance of both SHA-256 and SHA-512.The prominent reason for SHA-512 being practised more is because of its collision resistance nature. The collision resistance is discussed in research paper [7]. We focus on SHA-2 family particularly SHA-512, in this hashing function we use 64 bit words generated from input message and hash constants in each round with the total of 80 rounds to generate hash

ORCID(s): 0000-0003-1669-3895 (L. D)

function.When comparing with the SHA-1 and SHA-256, SHA-512 takes more time to execute its function because of large buffer size and more number of rounds. More number of rounds increases the security at the same time decreases the performance of an algorithm. In this paper we have tried to reduce the rounds by adding more hash functions in each round and final hash is generated after 40^{th} round. Benefits of reducing the number of rounds will increase the performance of an algorithm , increases the program efficiency , reduces the loop overhead , optimize the execution time of program , allocating buffers 80 times for 80 rounds will be reduced to 40. The complete procedure of SHA-512 and MSHA-512 have been discussed. The performance comparison have been discussed in the result section.

2. Algorithm Parameters & Symbols

2.1. Symbols used in algorithm

The following symbols are used in the secure hash algorithm specifications, and each operates on 64-bit words.

1. $\wedge$ Bitwise AND operation.
2. $\vee$ Bitwise OR (inclusive-OR) operation.
3. $\oplus$ Bitwise XOR (exclusive-OR) operation.
4. $\neg$ Bitwise complement operation.
5. $+$ Addition modulo 2^w.($w = 64$ bit as the word size is 64 bit in both SHA-512 and MSHA-512).
6. $\ll$ Left-shift operation, where w $\ll$ n is obtained by discarding the left-most n bits of the word w and then padding the result with n zeroes on the right.
7. $\gg$ Right-shift operation, where w $\gg$ n is obtained by discarding the right-most n bits of the word w and then padding the result with n zeroes on the left.

2.2. Parameters

1. a, b, c, ..., h Working variables that are the w-bit words used in the computation of the hash values, H(i)
2. N Number of blocks in the padded message.
3. M Message to be hashed
4. K_t Constant value to be used for iteration t of the hash computation.
5. ℓ Length of the message M, in bits.

3. SHA-512 Standard hash

Table 1
SHA-2 Family

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Rounds
SHA-256	$< 2^{64}$	512	32	256	64
SHA-384	$< 2^{128}$	1024	64	384	80
SHA-512	$< 2^{128}$	1024	64	512	80

3.1. SHA-512 Functions & Constants

3.1.1. Functions

1. Ch(x,y,z) = $(x \wedge y) \oplus (\neg x \wedge z)$
2. Maj(x,y,z) = $(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
3. $\sum_0^{512} = \mathrm{ROTR}^{28}(x) \oplus \mathrm{ROTR}^{34}(x) \oplus \mathrm{ROTR}^{39}(x)$
4. $\sum_1^{512} = \mathrm{ROTR}^{14}(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{ROTR}^{41}(x)$
5. $W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$
6. $\sigma_0^{512} = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x)$
7. $\sigma_1^{512} = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x)$

```
k[0..79] := [ 0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc, 0x3956c25bf348b538,
              0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118, 0xd807aa98a3030242, 0x12835b0145706fbe,
              0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2, 0x72be5d74f27b896f, 0x80deb1fe3b1696b1, 0x9bdc06a725c71235,
              0xc19bf174cf692694, 0xe49b69c19ef14ad2, 0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
              0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4, 0x76f988da831153b5, 0x983e5152ee66dfab,
              0xa831c66d2db43210, 0xb00327c898fb213f, 0xbf597fc7beef0ee4, 0xc6e00bf33da88fc2, 0xd5a79147930aa725,
              0x06ca6351e003826f, 0x142929670a0e6e70, 0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed,
              0x53380d139d95b3df, 0x650a73548baf63de, 0x766a0abb3c77b2a8, 0x81c2c92e47edaee6, 0x92722c851482353b,
              0xa2bfe8a14cf10364, 0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30, 0xd192e819d6ef5218,
              0xd69906245565a910, 0xf40e35855771202a, 0x106aa07032bbd1b8, 0x19a4c116b8d2d0c8, 0x1e376c085141ab53,
              0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8, 0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373,
              0x682e6ff3d6b2b8a3, 0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72, 0x8cc702081a6439ec,
              0x90befffa23631e28, 0xa4506cebde82bde9, 0xbef9a3f7b2c67915, 0xc67178f2e372532b, 0xca273eceea26619c,
              0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178, 0x06f067aa72176fba, 0x0a637dc5a2c898a6,
              0x113f9804bef90dae, 0x1b710b35131c471b, 0x28db77f523047d84, 0x32caab7b40c72493, 0x3c9ebe0a15c9bebc,
              0x431d67c49c100d4c, 0x4cc5d4becb3e42b6, 0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817]
```

Figure 1: Constants used in the Secure hash algorithm

3.1.2. Constants

SHA-512 uses the sequence of 80 constant 64-bit words , $K_0^{512}, K_1^{512}, K_2^{512}, ..., K_{79}^{512}$. These constants are formed by the fractional part of cube roots of the first eighty prime numbers.The Constants represented in the Figure1are in Hexadecimal (from left to right).

3.1.3. Initialization of vectors

For SHA-512, Initialization vectors, in hex:

1. $H_0^{(0)}$ = 0x6a09e667f3bcc908
2. $H_4^{(0)}$ = 0x510e527fade682d1
3. $H_1^{(0)}$ = 0xbb67ae8584caa73b16
4. $H_5^{(0)}$ = 0x9b05688c2b3e6c1f
5. $H_2^{(0)}$ = 0x3c6ef372fe94f82b
6. $H_6^{(0)}$ = 0x1f83d9abfb41bd6b
7. $H_3^{(0)}$ = 0xa54ff53a5f1d36f1
8. $H_7^{(0)}$= 0x5be0cd19137e2179

These words are obtained by drawing the fractional part of square roots of the first eight prime numbers.

3.2. Propagating Hash values

SHA-512 Algorithm can be obtained from two methods : **Pre-Processing** and **hash computation**.

3.2.1. Pre-Processing

The process includes padding of message.The padded message is divided into m-bit blocks which are used for hash computation.The first hash is generated for block-1 using initial vector values.The process will be continued for remaining blocks by using previous generated hash value.

Appending the bit “1” to the end of the message. For example Message(8-bit ASCII) “abc” has length $8 \times 3 = 24$,so the message is padded with a “1” bit , then $896-(24+1) = 871$ “0” bits and then the message length(ℓ), to become the 1024- bit padded message.

3.2.2. Word Generation

The first 16 words fromW_0 to W_{15} are fetched from the message block of size 1024 bits. Remaining words from W_{16} to $W_{79}$are calculated from the formula 5 in 3.1.1 section.

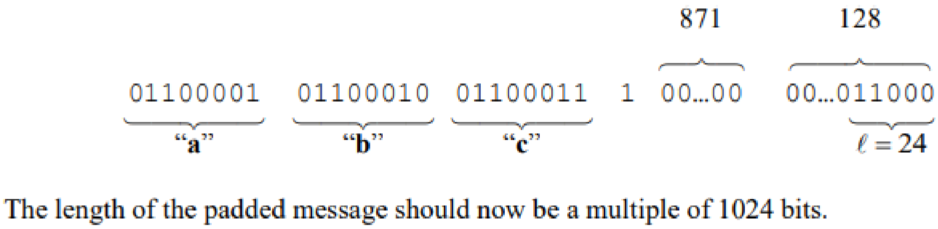

Figure 2: 1024 bits message block

3.3. SHA-512 Hash Computation

The padded message of 128 bit length is a complete formatted input, this input is divided into different blocks of size 1024 bits. Each block acts as message schedule which consists of 80 words ($W_0 to W_{79}$) ,each word of size 64 bit. Hash is generated by applying SHA-512 hash functions using Words and constants. The procedure is repeated for all the blocks to generate final hash.

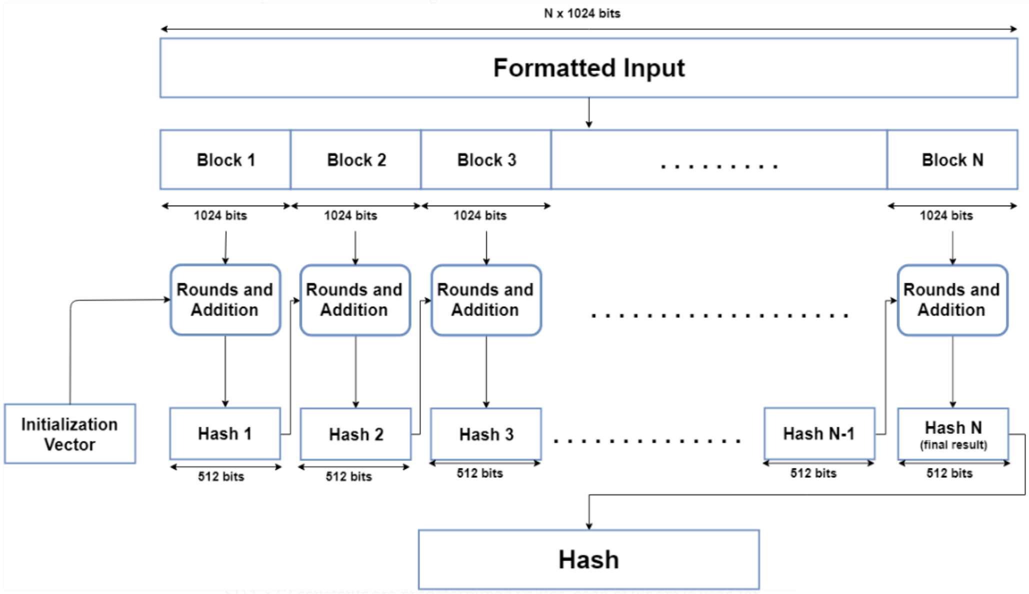

Figure 3: 1024 bits message block

3.3.1. SHA-512 Hash Propagation using 80 rounds (Rounds and Addition)

SHA-512 consists of 80 rounds, in each round hash function is used to generate hash for the next round which is stored in 8 buffers as shown in the figure 4. In each round, Word W_i and hash constant K_i is used to generate intermediate hash. After completing 80 rounds final result is added with the Initial vectors to get final hash message.

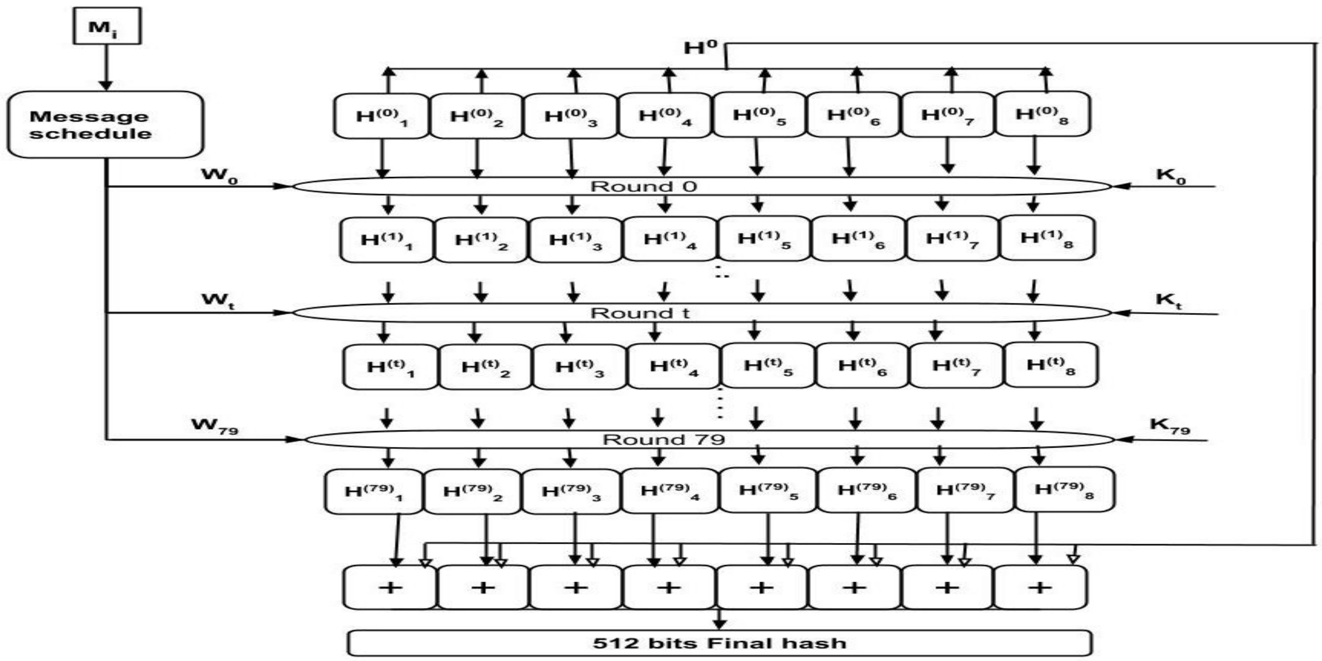

Figure 4: SHA-512 Hash generator using 80 rounds

3.3.2. SHA-512 Round function

Initially 8 buffers are assigned with initial vectors 3.1.3.In each round from 0 to 79, buffer values will get updated and which is calculated by the round functions. Round functions are referred from 1, 2 ,3 and 4. These functions are applied as shown in the Figure 5.

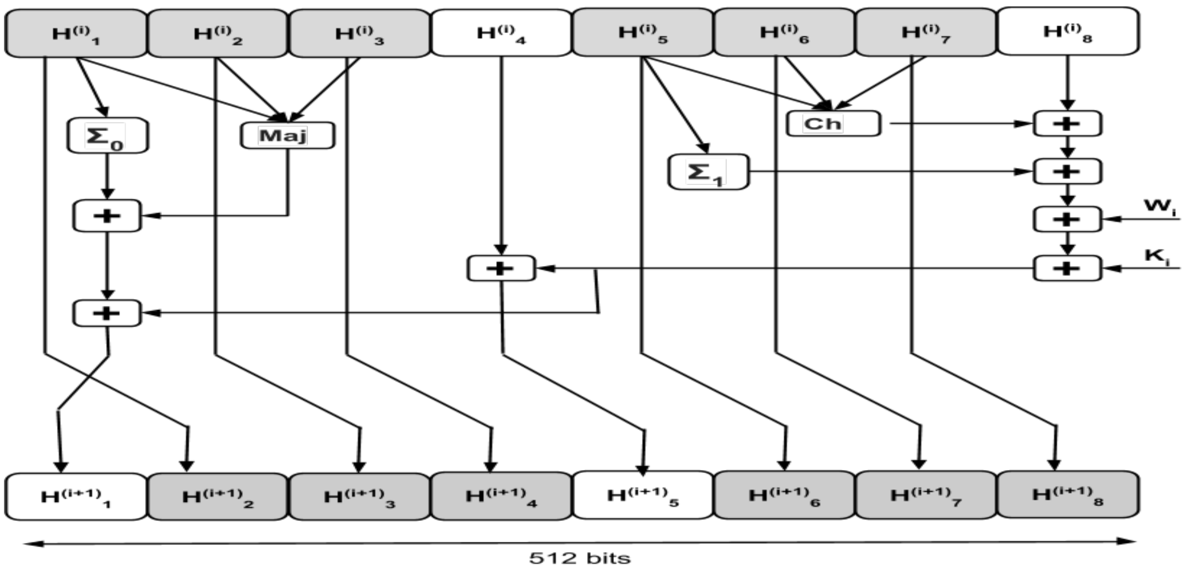

Figure 5: SHA-512 Round function for each round

The final hash propagated after applying round function is shown below for message “abc”:
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

4. Implementation of MSHA-512

MSHA-512 is implemented in order to increase the performance of the SHA-512 algorithm. We propose a new hash function which will reduce the rounds of SHA-512 to its half. In SHA-512 we need minimum 80 rounds to generate the hash of any message but in MSHA-512 40 rounds are enough to generate exactly the same hash for the same message.
For example as we discussed in the SHA-512 for message “abc” hash generated was :
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

The above hash was generated by applying the SHA-512 round functions to all 80 rounds. Similarly for the same message “abc” hash is generated using MSHA-512 is :
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

The above hash was generated by applying the MSHA-512 round functions in to 40 rounds. More rounds increases the security in cryptography but it also decreases the performance therefore the aim of MSHA-512 is to increase the performance of the algorithm by reducing the rounds and the same security is maintained by applying the new hash functions in MSHA-512.

4.1. MSHA-512 hash computation (Reducing rounds)

In MSHA-512 hash is generated after completion of 40 rounds. In each round we will use two consecutive words and two consecutive constants to generate hash by using round function.Obtained hash is used in next rounds for generating final hash.Reference 1 constants are used in this process.

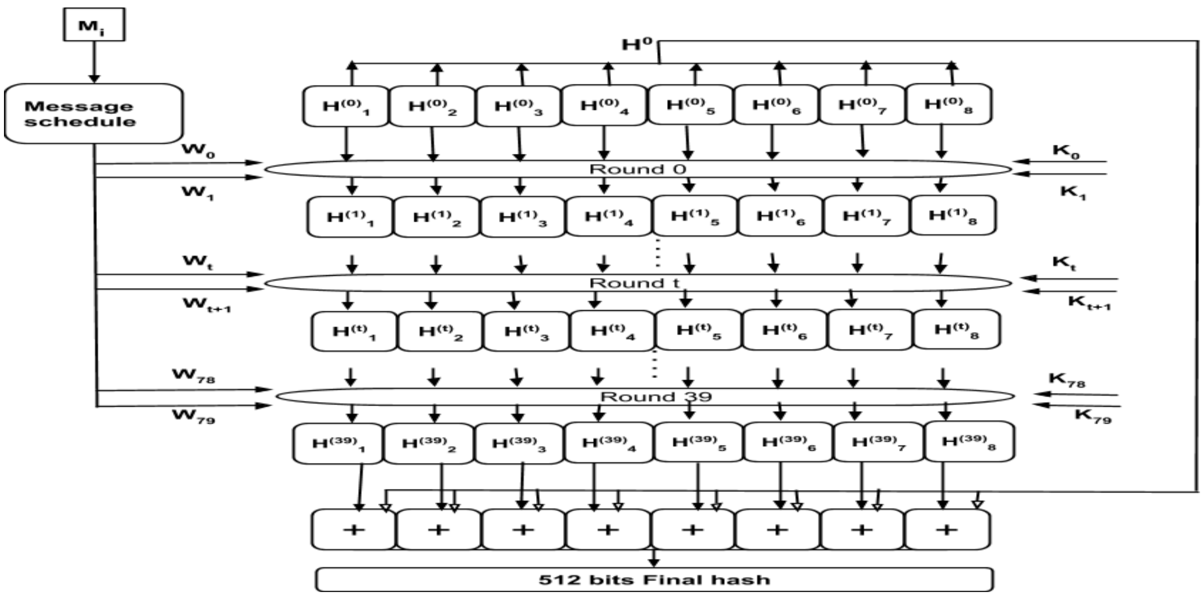

Figure 6: MSHA-512 hash computation

4.1.1. Hash computation procedure

Eight buffers are stored using initial vectors $H_0^{(0)}, H_1^{(0)}, H_2^{(0)}, H_3^{(0)}, H_4^{(0)}, H_5^{(0)}, H_6^{(0)}, H_7^{(0)}$ reference 3.1.3. Each vector named as a,b,c,d,e,f,g,h respectively. In each round these eight buffers are used to store intermediate result.

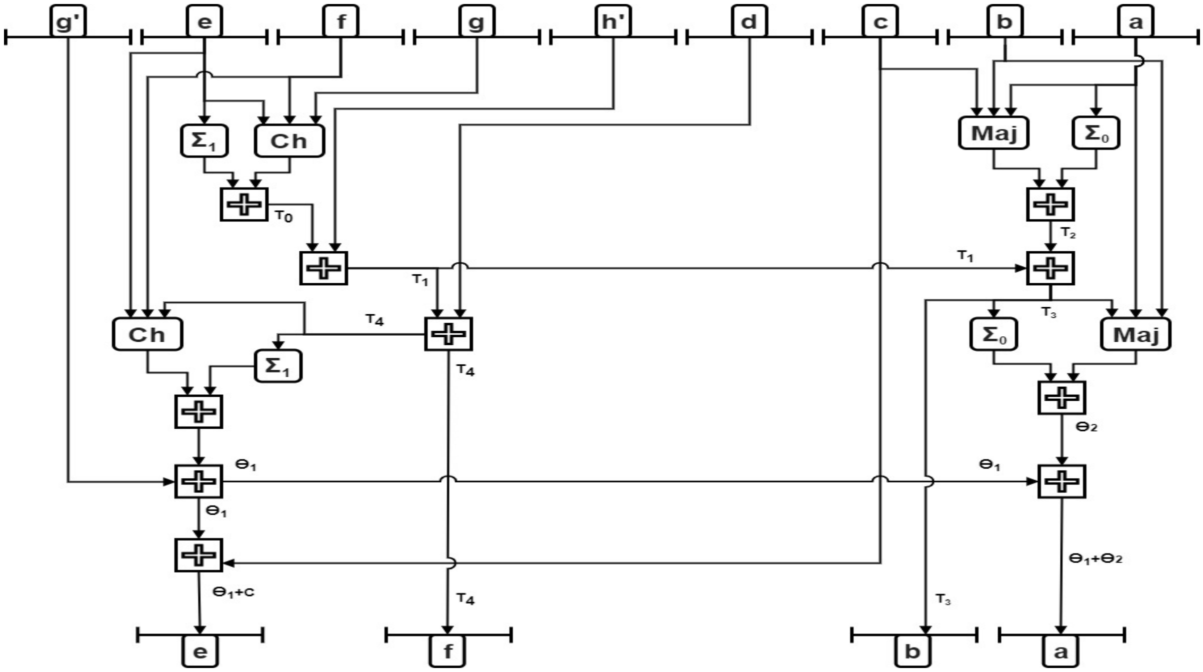

Figure 7: Message compression technique for MSHA-512

Where g' = $g+W_{t+1}+K_{t+1}$ h' $=h+W_t+K_t$

From round 0 to 39, buffer value will get updated and these buffer values are calculated by the round functions 1, 2 ,3 and 4 referred from section 3.1.1. These functions are applied as shown in the Figure 7 and 8 .

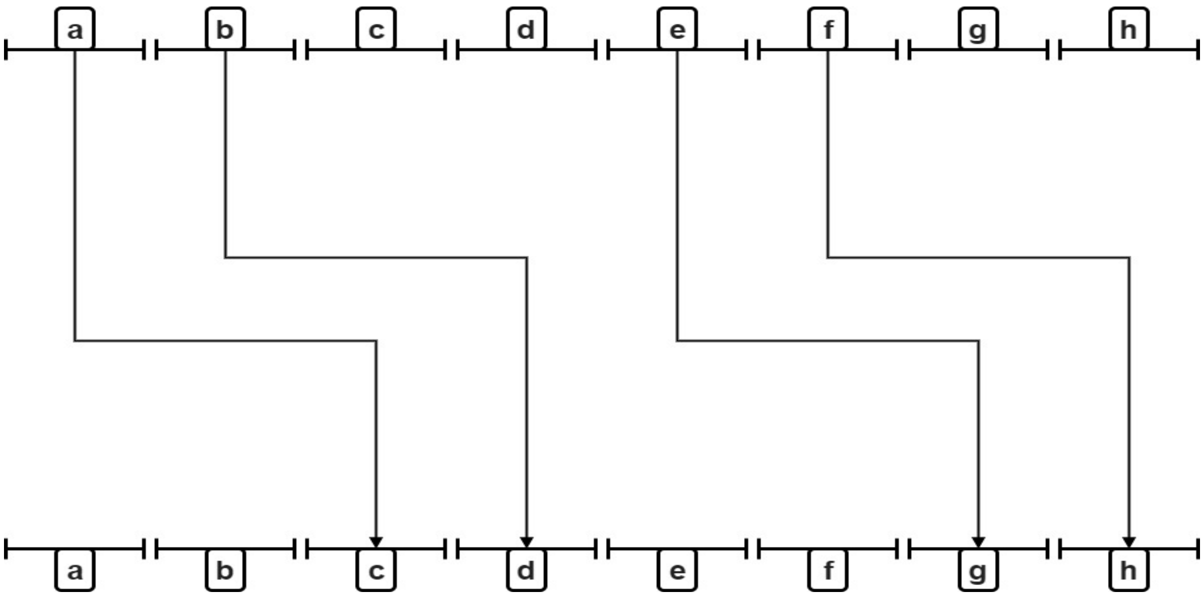

Figure 8: Buffer Swapping

MSHA-512 hash computation procedure involves two methods in each round.

- Message compression technique.
- Buffer swapping.

4.1.2. MSHA-512 Hash Computation algorithm

MSHA-512 uses both message compression technique and buffer swapping method in each round as shown in algorithm 4.1.2 to generate final hash.

For i = 1 to N

{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{512\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 79 \end{cases}$$

2. Initialize the eight working variables, $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$, with the $(i\text{-}1)^{st}$ hash value:

$$\begin{aligned}
a &= H_0^{(i-1)} \\
b &= H_1^{(i-1)} \\
c &= H_2^{(i-1)} \\
d &= H_3^{(i-1)} \\
e &= H_4^{(i-1)} \\
f &= H_5^{(i-1)} \\
g &= H_6^{(i-1)} \\
h &= H_7^{(i-1)}
\end{aligned}$$

3. **for (t = 0 ; t < 80 ; t = t+2)**

{

$$\begin{aligned}
h' &= h + W_t + K_t \\
g' &= g + W_{t+1} + K_{t+1} \\
T_1 &= \sum_1^{512}(e) + ch(e, f, g) + h' \\
T_2 &= \sum_0^{512}(a) + Maj(a, b, c) \\
T_3 &= T_1 + T_2 \\
T_4 &= T_1 + d \\
\Theta_1 &= \sum_1^{512}(T_4) + ch(T_4, e, f) + g' \\
\Theta_2 &= \sum_0^{512}(T_3) + Maj(T_3, a, b) \\
h &= f \\
g &= e \\
f &= T_4 \\
e &= c + \Theta_1 \\
d &= b \\
c &= a \\
b &= T_3 \\
a &= \Theta_1 + \Theta_2
\end{aligned}$$

}

4. **Compute the i_{th} intermediate hash value $H^{(i)}$:**

$$H_0^{(i)} = a + H_0^{(i-1)} \qquad H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)} \qquad H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)} \qquad H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)} \qquad H_7^{(i)} = h + H_7^{(i-1)}$$

}

Figure 9: MSHA-512 Hash Computation algorithm

4.2. MSHA-512 example (one-block Message)

4.2.1. Preprocessing - Message scheduling

Preprocessing involves padding message , initialize buffer , assigning the constants. Let the message 'M' of ASCII string be "abc" of 24-bit (ℓ=24) , 24-bit binary string : 01100001 01100010 01100011 which is equivalent to 616263 in hex value The length of string 'M' is 24 bits which is equivalent to 18 hex value The message is padded by appending a "1" bit, followed by 871 "0" bits, and ending with the hex value 0000000000000000 0000000000000018.
The 1024-bit message block, in hexadecimal is,

6162638000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000018

First 16 Words are generated from message block that is from W_0 to W_{15}
W_0 = 6162638000000000 W_1 = 0000000000000000 W_2 = 0000000000000000 W_3 = 0000000000000000
W_4 = 0000000000000000 W_5 = 0000000000000000 W_6 = 0000000000000000 W_7 = 0000000000000000
W_8 = 0000000000000000 W_9 = 0000000000000000 W_{10} = 0000000000000000 W_{11} = 0000000000000000
W_{12} = 0000000000000000 W_{13} = 0000000000000000 W_{14} = 0000000000000000 W_{15} = 0000000000000018
MSHA-512 needs 80 words for processing and the remaining words from W_{16} to W_{79} is calculated using formula reference 5 in section 3.1.1
MSHA-512 constants are drawn from SHA-512, reference 1.

4.2.2. Hash generation

The following schedule shows the hex values for a, b, c, d, e, f, g, and h after pass t of the "for t = 0 to 39" loop described in Figure 7 and 8 .

Table 2
Hash table for 40 rounds

Rounds	a/e	b/f	c/g	d/h
t=0	1320f8c9fb872cc0 c3d4ebfd48650ffa	f6afceb8bcfcddf5 58cb02347ab51f91	6a09e667f3bcc908 510e527fade682d1	bb67ae8584caa73b 9b05688c2b3e6c1f
t=1	5a83cb3e80050e82 0b47b4bb1928990e	ebcffc07203d91f3 dfa9b239f2697812	1320f8c9fb872cc0 c3d4ebfd48650ffa	f6afceb8bcfcddf5 58cb02347ab51f91
t=2	af573b02403e89cd 96f60209b6dc35ba	b680953951604860 745aca4a342ed2e2	5a83cb3e80050e82 0b47b4bb1928990e	ebcffc07203d91f3 dfa9b239f2697812
t=3	8093d195e0054fa3 86f67263a0f0ec0a	c4875b0c7abc076b 5a6c781f54dcc00c	af573b02403e89cd 96f60209b6dc35ba	b680953951604860 745aca4a342ed2e2
t=4	81782d4a5db48f03 00091f460be46c52	f1eca5544cb89225 d0403c398fc40002	8093d195e0054fa3 86f67263a0f0ec0a	c4875b0c7abc076b 5a6c781f54dcc00c
t=5	db0a9963f80c2eaa 475975b91a7a462c	69854c4aa0f25b59 d375471bde1ba3f4	81782d4a5db48f03 00091f460be46c52	f1eca5544cb89225 d0403c398fc40002
t=6	44249631255d2ca0 860acf9effba6f61	5e41214388186c14 cdf3bff2883fc9d9	db0a9963f80c2eaa 475975b91a7a462c	69854c4aa0f25b59 d375471bde1ba3f4

Rounds	a/e	b/f	c/g	d/h
t=7	0ae07c86b1181c75 a77b7c035dd4c161	fa967eed85a08028 874bfe5f6aae9f2f	44249631255d2ca0 860acf9effba6f61	5e41214388186c14 cdf3bff2883fc9d9
t=8	4725be249ad19e6b f47e8353f8047455	caf81a425d800537 2deecc6b39d64d78	0ae07c86b1181c75 a77b7c035dd4c161	fa967eed85a08028 874bfe5f6aae9f2f
t=9	9a3fb4d38ab6cf06 f14998dd5f70767e	3c4b4104168e3edb 29695fd88d81dbd0	4725be249ad19e6b f47e8353f8047455	caf81a425d800537 2deecc6b39d64d78
t=10	da34d6673d452dcf 8e30ff09ad488753	8dc5ae65569d3855 4bb9e66d1145bfdc	9a3fb4d38ab6cf06 f14998dd5f70767e	3c4b4104168e3edb 29695fd88d81dbd0
t=11	4f6877b58fe55484 c66005f87db55233	3e2644567b709a78 0ac2b11da8f571c6	da34d6673d452dcf 8e30ff09ad488753	8dc5ae65569d3855 4bb9e66d1145bfdc
t=12	0bc5f791f8e6816b 6ddf1fd7edcce336	9aff71163fa3a940 d3ecf13769180e6f	4f6877b58fe55484 c66005f87db55233	3e2644567b709a78 0ac2b11da8f571c6
t=13	eab4a9e5771b8d09 09068a4e255a0dac	884c3bc27bc4f941 e6e48c9a8e948365	0bc5f791f8e6816b 6ddf1fd7edcce336	9aff71163fa3a940 d3ecf13769180e6f
t=14	74bf40f869094c63 f0aec2fe1437f085	e62349090f47d30a 0fcdf99710f21584	eab4a9e5771b8d09 09068a4e255a0dac	884c3bc27bc4f941 e6e48c9a8e948365
t=15	ff4d3f1f0d46a736 3cd388e119e8162e	4c4fbbb75f1873a6 73e025d91b9efea3	74bf40f869094c63 f0aec2fe1437f085	e62349090f47d30a 0fcdf99710f21584
t=16	60d4e6995ed91fe6 efabbd8bf47c041a	a0509015ca08c8d4 e1034573654a106f	ff4d3f1f0d46a736 3cd388e119e8162e	4c4fbbb75f1873a6 73e025d91b9efea3
t=17	1a081afc59fdbc2c f098082f502b44cd	2c59ec7743632621 0fbae670fa780fd3	60d4e6995ed91fe6 efabbd8bf47c041a	a0509015ca08c8d4 e1034573654a106f
t=18	002bb8e4cd989567 66adcfa249ac7bbd	88df85b0bbe77514 8fbfd0162bbf4675	1a081afc59fdbc2c f098082f502b44cd	2c59ec7743632621 0fbae670fa780fd3
t=19	8e01e125b855d225 0c710a47ba6a567b	b3bb8542b3376de5 b49596c20feba7de	002bb8e4cd989567 66adcfa249ac7bbd	88df85b0bbe77514 8fbfd0162bbf4675
t=20	e96f89dd48cbd851 f0996439e7b50cb1	b01521dd6a6be12c 169008b3a4bb170b	8e01e125b855d225 0c710a47ba6a567b	b3bb8542b3376de5 b49596c20feba7de
t=21	35d7e7f41defcbd5 cc5100997f5710f2	bc05ba8de5d3c480 639cb938e14dc190	e96f89dd48cbd851 f0996439e7b50cb1	b01521dd6a6be12c 169008b3a4bb170b
t=22	021fbadbabab5ac6 e95c2a57572d64d9	c47c9d5c7ea8a234 858d832ae0e8911c	35d7e7f41defcbd5 cc5100997f5710f2	bc05ba8de5d3c480 639cb938e14dc190
t=23	6b69fc1bb482feac 35264334c03ac8ad	f61e672694de2d67 c6bc35740d8daa9a	021fbadbabab5ac6 e95c2a57572d64d9	c47c9d5c7ea8a234 858d832ae0e8911c
t=24	ca9bd862c5050918 dfe091dab182e645	571f323d96b3a047 271580ed6c3e5650	6b69fc1bb482feac 35264334c03ac8ad	f61e672694de2d67 c6bc35740d8daa9a
t=25	d43f83727325dd77 483f80a82eaee23e	813a43dd2c502043 07a0d8ef821c5e1a	ca9bd862c5050918 dfe091dab182e645	571f323d96b3a047 271580ed6c3e5650
t=26	d63f68037ddf06aa a6781efe1aa1ce02	03df11b32d42e203 504f94e40591cffa	d43f83727325dd77 483f80a82eaee23e	813a43dd2c502043 07a0d8ef821c5e1a
t=27	63b460e42748817e c6b4dd2a9931c509	f650857b5babda4d 9ccfb31a86df0f86	d63f68037ddf06aa a6781efe1aa1ce02	03df11b32d42e203 504f94e40591cffa

Rounds	a/e	b/f	c/g	d/h
t=28	4b81c3aec976ea4b 70505988124351ac	7a52912943d52b05 d2e89bbd91e00be0	63b460e42748817e c6b4dd2a9931c509	f650857b5babda4d 9ccfb31a86df0f86
t=29	2c074484ef1eac8c 4797cde4ed370692	581ecb3355dcd9b8 6a3c9b0f71c8bf36	4b81c3aec976ea4b 70505988124351ac	7a52912943d52b05 d2e89bbd91e00be0
t=30	cfcd928c5424e2b6 09aee5bda1644de5	3857dfd2fc37d3ba a6af4e9c9f807e51	2c074484ef1eac8c 4797cde4ed370692	581ecb3355dcd9b8 6a3c9b0f71c8bf36
t=31	ab44e86276478d85 cd881ee59ca6bc53	a81dedbb9f19e643 84058865d60a05fa	cfcd928c5424e2b6 09aee5bda1644de5	3857dfd2fc37d3ba a6af4e9c9f807e51
t=32	eeb9c21bb0102598 3b5fed0d6a1f96e1	5a806d7e9821a501 aa84b086688a5c45	ab44e86276478d85 cd881ee59ca6bc53	a81dedbb9f19e643 84058865d60a05fa
t=33	54ba35cf56a0340e 1c66f46d95690bcf	46c4210ab2cc155d 29fab5a7bff53366	eeb9c21bb0102598 3b5fed0d6a1f96e1	5a806d7e9821a501 aa84b086688a5c45
t=34	fb6aaae5d0b6a447 e3711cb6564d112d	181839d609c79748 0ada78ba2d446140	54ba35cf56a0340e 1c66f46d95690bcf	46c4210ab2cc155d 29fab5a7bff53366
t=35	f15e9664b2803575 947c3dfafee570ef	7652c579cb60f19c aff62c9665ff80fa	fb6aaae5d0b6a447 e3711cb6564d112d	181839d609c79748 0ada78ba2d446140
t=36	20878dcd29cdfaf5 054d3536539948d0	358406d165aee9ab 8c7b5fd91a794ca0	f15e9664b2803575 947c3dfafee570ef	7652c579cb60f19c aff62c9665ff80fa
t=37	c8960e6be864b916 995019a6ff3ba3de	33d48dabb5521de2 2ba18245b50de4cf	20878dcd29cdfaf5 054d3536539948d0	358406d165aee9ab 8c7b5fd91a794ca0
t=38	d67806db8b148677 25c96a7768fb2aa3	654ef9abec389ca9 ceb9fc3691ce8326	c8960e6be864b916 995019a6ff3ba3de	33d48dabb5521de2 2ba18245b50de4cf
t=39	73a54f399fa4b1b2 d08446aa79693ed7	10d9c4c4295599f6 9bb4d39778c07f9e	d67806db8b148677 25c96a7768fb2aa3	654ef9abec389ca9 ceb9fc3691ce8326

The Process will completed for one message block $M^{(1)}$,if we have 'N' message blocks $M^{(N)}$ each message block will be processed 40 rounds as shown in figure 3. After 40^{th} round the final hash is generated by adding Hash 'N' and initial vectors.

$H_0^{(0)}$ = 0x6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba
$H_1^{(0)}$ = 0xbb67ae8584caa73b16 + 10d9c4c4295599f6 = cc417349ae204131
$H_2^{(0)}$ = 0x3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2
$H_3^{(0)}$ = 0xa54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a
$H_4^{(0)}$ = 0x510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8
$H_5^{(0)}$ = 0x9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd
$H_6^{(0)}$ = 0x1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e
$H_7^{(0)}$= 0x5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f

The resulting 512-bit message digest for string message "abc",
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a
2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

5. Results

5.1. Hash comparison

Experimental result shows that the final hash generated in the 40^{th} round of MSHA-512 is similar to the final hash generated by SHA-512.

Hash generated in SHA-512 [3]	Hash generated in MSHA-512
ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a 2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f	ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a 2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

Table 4
Hash generated in both SHA-512 and MSHA-512

5.2. Features of SHA-512 and MSHA-512

By comparing the features of SHA-512 and MSHA-512 algorithm we can observe that the time consumed for processing is less in MSHA-512 and also the performance rate is higher when compared to SHA-512 algorithm.

FEATURES	SHA-512	MSHA-512
Clock Cycles	80	40
MaxFrequency (MHz)	118.043MHz	118.043MHz
Throughput	1.51Gbps	3.02 Gbps
Digest Length	512	512
Time delay (ns)	677.719ns	314.931ns

Table 5
Features of SHA-512 and MSHA-512

5.3. Throughput

The Maximum data throughput can be computed by the following equation:

$$Throughput = \frac{Message\ Block\ Size * Max\ Clock\ Frequency}{Number\ of\ Rounds} \tag{1}$$

$$Max\ Clock\ Frequency = \frac{1}{(Max\ Data\ Path\ Delay - Min\ Data\ Path\ Delay + Tsetup)} \tag{2}$$

For one message block of ASCII string “abc”.
Message Block size = 1024 bits.
Max clock Frequency of SHA-512 is 118.043MHz from the timing analysis.
Assuming that we have Max Clock Frequency of MSHA-512 is 118.043MHz. Even though frequency will be less than that of SHA-512. Below throughput can be drawn using above formula.
Using the (eq. 1) maximum expected throughput for SHA-512 is: **1024b × 118.043MHz80= 1.51Gbps.**
Using the (eq. 1) maximum expected throughput for MSHA-512 : **1024b × 118.043MHz40= 3.02Gbps**

5.4. Performance comparison of SHA-512 and MSHA-512

To measure the performance of SHA-512 and MSHA-512 we have implemented in JavaScript, for easy access and quick generation of hash particularly in Blockchain technology.The average time for generating hash in both SHA-512 and MSHA-512 is calculated and the timing graph is generated based on the performance.The timing graph shown below represents the time taken for hash generation in both SHA-512 and MSHA-512 for one block message of ASCII string "abc".

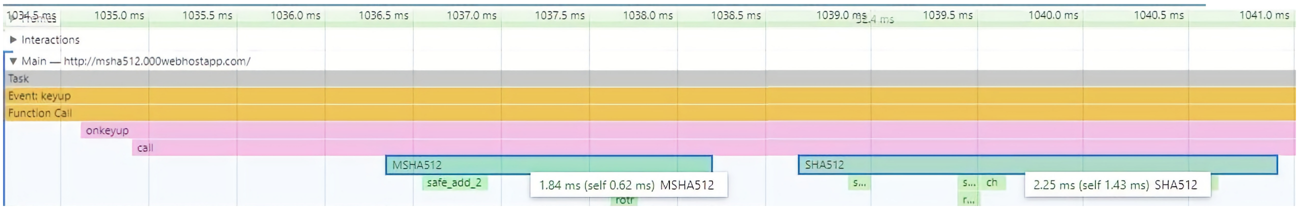

Figure 10: Execution time of an both SHA-512 and MSHA-512 using web servers

As shown in the figure 10 timing graph of MSHA-512 algorithm yields 0.62ms to execute the function and the total time taken to generate hash is 1.84ms, similarly SHA-512 algorithm yields 1.43ms to execute function and the total time taken to generate hash is 2.25ms.
The Time comparison between MSHA-512 and SHA-512 is shown below :

MSHA-512 hash generation time			SHA-512 hash generation time		
Self Time	Total Time	Activity	Self Time	Total Time	Activity
0.6 ms 33.8 %	1.8 ms 100.0 %	MSHA512	1.4 ms 63.6 %	2.3 ms 100.0 %	SHA512
0.5 ms 27.5 %	0.5 ms 27.5 %	safe_add_2	0.2 ms 10.6 %	0.2 ms 10.6 %	safe_add_4
0.4 ms 24.0 %	0.4 ms 24.0 %	maj	0.2 ms 10.4 %	0.2 ms 10.4 %	safe_add_5
0.2 ms 8.6 %	0.2 ms 8.6 %	rotr	0.1 ms 5.3 %	0.1 ms 5.3 %	ch
0.1 ms 6.2 %	0.1 ms 6.2 %	binb2hex	0.1 ms 5.1 %	0.2 ms 10.1 %	sigma1
			0.1 ms 5.1 %	0.1 ms 5.1 %	rotr

Table 6
Functions time comparison

6. Conclusion

SHA-512 is a powerful hash function which generates hash with high collision resistance feature. Because of its message compression technique and 80 rounds to generate hash provides us more security for the different kind of applications like Integrity maintenance, message authentication and Blockchain technology. We have presented MSHA-512 with high performance having 40 rounds instead of 80 rounds when compared with SHA-512 algorithm.Result shows that MSHA-512 produces less clock cycles and less frequency with high throughput. The experimental result shows that generation of hash for particular string is same in both SHA-512 and MSHA-512.The Results also discussed that MSHA-512 is more powerful than SHA-512 in both performance as well as in security. The code was tested and successfully implemented in JavaScript. The Google development tools are used to analyse and compare both SHA-512 and MSHA-512 functions. MSHA-512 in future helps in generating hash for blockchain with quick access and fast retrieval of blocks and we can merge blockchain with IOT for better communication process.

References

[1] Rivest, R. L. The MD5 Message Digest Algorithm. IETF Network Working Group, RFC 1321, 1992.

[2] "Fast MD5 and MD4 Collision Generators". Retrieved 10 February 2014. Faster implementation of techniques in How to Break MD5 and Other Hash Functions, by Xiaoyun Wang, et al April 2006

[3] Federal Information Processing Standards Publication 180-3, "SECURE HASH STANDARD", October 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

[4] Marc Stevens1 et al. "The first collision for full SHA-1" https://shattered.io

[5] Ryan Glabb et al. "Multi-mode operator for SHA-2 hash functions".

[6] N. SKLAVOS et al ."Implementation of the SHA-2 Hash Family Standard Using FPGAs"

[7] Christoph Dobraunig, et al. "Analysis of SHA-512/224 and SHA-512/256".