Journal of Physics:
Conference Series



**PAPER** • **OPEN ACCESS**

# A Comparative Analysis of the Application of Hashing Encryption Algorithms for MD5, SHA-1, and SHA-512

To cite this article: Sihan Long 2019 *J. Phys.: Conf. Ser.* **1314** 012210



IOP Conf. Series: Journal of Physics: Conf. Series **1314** (2019) 012210 doi:10.1088/1742-6596/1314/1/012210

# A Comparative Analysis of the Application of Hashing Encryption Algorithms for MD5, SHA-1, and SHA-512

**Sihan Long**

Changzhou University Huaide College, Taizhou, Jiangsu, 214500, China

Corresponding author's e-mail: angela@cas-harbour.org

**Abstract.** This study compares different hashing encryption algorithms and provides several suggestions about how to choose different hashing encryption algorithms for particular cases. This paper presents three different kinds of hashing encryption algorithms. The author introduces the fundamental conceptions of these algorithms and their application. The author also computes some significant factors in computer science and compares these algorithms by mathematical quantity computing data. This article discusses the advantages and disadvantages of hashing encryption algorithms and proposes its potential value in the future.

## 1. Introduction

Hashing is the key that converts a string to a fixed length value that is usually shorter or represents the original string. Hashing is used to find and retrieve items in a database. The efficiency of hashing greatly shortens the working process. For example, many encryption algorithms use a shorter hash key to find items instead of using the original value to find it in order to save time and energy[1-3]. Hashing scatters the records throughout the hash table in a completely random manner. This is the reason why hashing is appropriate for implementing a relationship among elements but it does not lend itself to operations, which attempts to make use of any other relationships among the data[4]. Hashing Encryption algorithms can apply to several aspects including MD5, SHA-1, SHA-512, etc. Surprisingly, people can use hashing encryption for different motivations. However, a hashing encryption algorithm is an irreversible algorithm. To be specific, it can be encrypted, but it cannot be deciphered. It also has valuable attributes in spite of its irreversible property such as file encryption and digital signature.

## 2. Technical Parts

### *2.1. MD5 algorithm*

#### *2.1.1. MD5 conception*

It is necessary to introduce the MD4 algorithm before relating the MD5 algorithm. The MD4 message digest algorithm uses any length of input messages and generates 128-bit output "fingerprints" or "message digests". In this way, it is not computationally possible to generate any messages with a given pre-specified target message digest. The MD4 algorithm is designed very fast on 32-bit machines [5].

It has six changes from MD4 algorithm:

- A fourth-round was added.
- The second round of function has been changed from most function to multiplexer function .

- The access order of the input words changed in the second and the third round.
- The number of shifts varies in each round. It is different now.
- Each step now has a unique addition constant.
- Now, each step adds the result of the previous step[6].

*2.1.2. MD5 algorithm procedure*

Step 1: Append padding bits

The messages are filled to match the length of 448 modules. This padding is to add a bit at the end of the message, followed by the required zero, so that the length of the bit is equal to 448 modulus 512.

Step 2: Append length

The message's length, which represents 64-bit, will be appended to the result. It makes the length of the message reaches exact multiple of 512 bits.

Step 3: Divide the message

Input the string in 512-bit blocks, and divided into 16*32-bit sub-blocks. The algorithm output is set of four 32-bit blocks; it forms a single 128-bit hash value.

Step 4: Initialize MD buffer

Four chaining variables are initialized.

Step 5: Process message in 16-blocks

The algorithm loop begins and continues when there are 512-bit blocks in the message. The main loop has four similar rounds. The result of coping four rounds into different chaining variables is illustrated as the following: a gets A, b gets B, c gets C, d gets D. Each operation performs a nonlinear function on three of a, b, c, and d. Then it adds the result to the right a variable number of bits and adds the result to one of a, b, c, and d. Finally, the result replaces one of a, b, c, and d[7].

Four nonlinear functions:

$$F(X,Y,Z) = (X \wedge Y) \vee ((\sim X) \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\sim Z))$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee (\sim Z))$$

Step 6: Output

Message digests generated as output are A, B, C, D. The output starts with low bytes of A and ends with high bytes of D.

*2.2. SHA-1 algorithm*

*2.2.1. SHA-1 algorithm conception*

SHA-1 is used to calculate the message digest of the message or data file provided as input. Messages or data files should be treated as a bit string. The length of the message is the number of bits in the message (the length of the empty message is 0). If the number of bits in the message is a multiple of 8, users can use hexadecimal to represent the message for compactness. The purpose of message filling is to multiply the total length of the filled message by 512[8].

*2.2.2. SHA-1 algorithm procedure*

Step 1: Initialize variables

The author presumes that the message length to be digested is fewer than $2^{64}$ .

$$h0 = 0x67452301$$

$$h1 = 0xEFCDAB89$$

$$h2 = 0x98BADCFE$$

$$h3 = 0x10325476$$

$$h4 = 0xC3D2E1F0$$

Step 2: Pre-processing

Append the bit "1" to the message, then append k bits "0", where k is the minimum number≥0, such as the resulting message length is congruent to 448 (mod 512). Append message length, in bits, as 64-bit big-endian integer.

Step 3: Process the message in consecutive 512-bit chunks

Break message into 512-bit chunks for each chunk, and the break chunk into sixteen 32-bit big-endian words. $w[i],\ 0 \pounds i \pounds 15$

Step 4: Extend the sixteen 32-bit words to eighty 32-bit words

$for\ i\ \ from\ 16\ to\ 79$

$w[i] = w[i-3] \oplus w[i-8] \oplus w[i-14] \oplus w[i-16]\ \ leftrotate\ \ 1$

Step 5: Initialize hash value for chunk

$a := h0$

$b := h1$

$c := h2$

$d := h3$

$e := h4$

Step 6: Main loop

$for\ i\ \ from\ 0\ \ to\ 79$

$if\ \ 0 \le i \le 19$ then

$f = (b \wedge c) \vee ((\neg b) \wedge d)$

$k = 0x5A827999$

$else\ if\ \ 20 \le i \le 39$

$f = b \oplus c \oplus d$

$k = 0x6ED9EBA1$

$else\ if\ \ 40 \le i \le 59$

$f = (b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$

$k = 0x8F1BBCDC$

$else\ if\ \ 60 \le i \le 79$

$f = b \oplus c \oplus d$

$k = 0xCA62C1D6$

$temp = (a\ \ leftrotate\ \ 5\ ) + f + e + k + w[i]$

$e = d$

$d = c$

$c = b\ \ leftrotate\ \ 30$

$b = a$

$a = temp$

Add the hash of this block to the result:

$h0 = h0 + a$

$h1 = h1 + b$

$h2 = h2 + c$

$h3 = h3 + d$

$h4 = h4 + e$

Produce the final hash value(big-endian):

$digest = hash = h0\ append\ h1\ append\ h2\ append\ h3\ append\ h4$

The above expressions for F are listed in FIPS PUB 180-1. The following alternative expressions may be able to calculate f in the main loop[9].

$(0 \le i \le 19)\ f = d \oplus (b \wedge (c \oplus d))\ (alternative)$

$(40 \le i \le 59)\ f = (b \wedge c) \vee (d \wedge (b \wedge c))\ (alternative\ 1)$

$(40 \le i \le 59)\ f = (b \wedge c) \vee (d \wedge (b \oplus c))\ (alternative\ 2)$

$(40 \le i \le 59)\ f = (b \wedge c) + (d \wedge (b \oplus c))\ (alternative\ 3)$

### *2.3. SHA-512 algorithm*

#### *2.3.1. SHA-512 algorithm conception*

"After introducing a new secret-key encryption standard, AES (Advanced Encryption Standard)" [10], with 128, 192, and 256-bit key sizes, the security of SHA-1 no longer meets the security guaranteed by encryption standards. Therefore, NSA has begun to develop three new hash functions, whose security matches the security of AES with 128, 192, and 256-bit keys, respectively. This work led to the publication of the draft Federal Information Processing Standard, and the introduction of three new hash functions, called SHA-256, SHA-384, and SHA-512[11]. The typical SHA-512 algorithm will be introduced in the following part.

#### *2.3.2. SHA-512 algorithm procedure*

Step 1: Initialize variables

The author presumes that the message length to be digested is fewer than $2^{128}$.

$H_0^0 \to a \quad 0x6a09e667f3bcc908$

$H_1^0 \to b \quad 0xbb67ae8584caa73b$

$H_2^0 \to c \quad 0x3c6ef372fe94f82b$

$H_3^0 \to d \quad 0xa54ff53a5f1d36f1$

$H_4^0 \to e \quad 0x510e527fade682dl$

$H_5^0 \to f \quad 0x9b05688c2b3e6c1f$

$H_6^0 \to g \quad 0x1f83d9abfb41bd6b$

$H_7^0 \to h \quad 0x5be0cd19137e2179$

Step 2: Pro-proceeding

(1) Append the bit "1" to the message, then append k bits "0", where k is the minimum number≥0, such as the resulting message length into multiple of 1024 bits.

(2) Parsing the padded message into N message blocks $B^0$, $B^1$,..., $B^N$, where the block size is 1024 bits[12].

Step 3 and Step 4 will be executed 80 times on the 1024-bit block.

Step 3: Prepare the message schedule

$$W_t = \begin{cases} B_t^i, (0 \le t \le 15) \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}, (16 \le t \le 80) \end{cases}$$

Where:

$\sigma_1 = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x),$

$\sigma_0 = ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x)$

Step 4: Initialize the eight working variables, a, b, c, d, e, f and h.

A sequence of 80 constant 64-bit words, are used by the processing unit[12].

$h = g,$
$g = f,$
$f = e,$
$e = d + T_1,$
$d = c,$
$c = b,$
$b = a,$
$a = T_1 + T_2.$

Where:

$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z),$
$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z),$
$\sum_0 (x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x),$
$\sum_1 (x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$

Step 5: Output

Assuming that the *ith* intermediate hash value is computed, the values from H0 to H7 are iterated and calculated by an adder.

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

After repeating the above steps N times, the result of message-digest is:

$H_0^N \| H_1^N \| H_2^N \| H_3^N \| H_4^N \| H_5^N \| H_6^N \| H_7^N .$

## 3. Simulation

Before the simulation, the author defines a conception that is the number of bytes encrypted per unit time (NBET). NBET is the number of bits of character/Average Running time. Firstly, this study compared the running time; the experiments were performed on Windows 10 with Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz 2.30GHz and 8.00GB RAM and 64-bit operating system. The running environment used for coding C++ in Visual Studio 2013. Each algorithm was tested to three character strings for six times. The results of running are illustrated in the next section.

### *3.1. MD5 Algorithm*

#### *3.1.1. First MD5 Algorithm character string test*

Character string: “abc”
Hash Value=900150983cd24fb0d6963f7d28e17f72
Average Running Time=3.08556 ms
NBET=0.97227 bit/ms

#### *3.1.2. Second MD5 Algorithm character string test*

Character string: “followingabcdefghijklmnopqrstuvwxyz”
Hash Value=21bb14b92f9d6f6205868eedeefcd4da
Average Running Time=5.37620 ms
NBET=6.51017 bit/ms

#### *3.1.3. Third MD5 Algorithm character string test*

Character string:
“qwertyuiopasdfghjklzxcvbnmpoiuytrewqasdfghjklmnbvcxzmnbvcxzasdfghjklpoiuytrewq”

Hash Value=779290ab2fb7ebc2aa91805df8559b39
Average Running Time=7.75992 ms
NBET=10.05168 bit/ms

*3.2. SHA-1 Algorithm*

*3.2.1. First SHA-1 Algorithm character string test*
Character string: "abc"
Hash Value=a9993e364706816aba3e25717850c26c9cd0d89d
Average Running Time=8.87031 ms
NBET=0.33821 bit/ms

*3.2.2. Second SHA-1 Algorithm character string test*
Character string: "followingabcdefghijklmnopqrstuvwxyz"
Hash Value=916e4febe8aae0e2438846d26db0dc2333f90ba6
Average Running Time=9.22622 ms
NBET=3.79353 bit/ms

*3.2.3. Third SHA-1 Algorithm character string test*
Character string:
"qwertyuiopasdfghjklzxcvbnmpoiuytrewqasdfghjklmnbvcxzmnbvcxzasdfghjklpoiuytrewq"
Hash Value=c200f0c3a9a671388a65c8f7546376cafa575fac
Average Running Time=15.60673 ms
NBET=4.99784 bit/ms

*3.3. SHA-512 Algorithm*

*3.3.1. First SHA-512 Algorithm character string test*
Character string: "abc"
Hash
Value=ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9eeee64b55d39a2192992a274fc1a8
36ba3c23a3feebbd454d4423643ce80e2a9ac94fa54ca49f
Average Running Time=10.79296 ms
NBET=0.27796 bit/ms

*3.3.2. Second SHA-512 Algorithm character string test*
Character string: "followingabcdefghijklmnopqrstuvwxyz"
Hash
Value=35146a6d3363d40006f4f09ccf06f5b804bc8c5cf8155ccbcf0f1e36b4a6b0cdc610b6b2890c4d48
bd5d2842c64c89e23725be66504a67779d78580e78edaf14
Average Running Time=13.27163 ms
NBET=2.63720 bit/ms

*3.3.3. Third SHA-512 Algorithm character string test*
Character string:
"qwertyuiopasdfghjklzxcvbnmpoiuytrewqasdfghjklmnbvcxzmnbvcxzasdfghjklpoiuytrewq"
Hash
Value=3d0efb042b7c59a39a7db85028b9ecc468a3b20a7f052e7b1c93689b3480f9c4c1a2ec8fac3c1cd7
9fe434c19a0cd01f724b49286456555f50c31a6f16ae35d0
Average Running Time=18.18712 ms
NBET=4.28875 bit/ms

*3.4. Comparison of Running Time of MD5 algorithm and SHA-1 algorithm and SHA-512 algorithm*

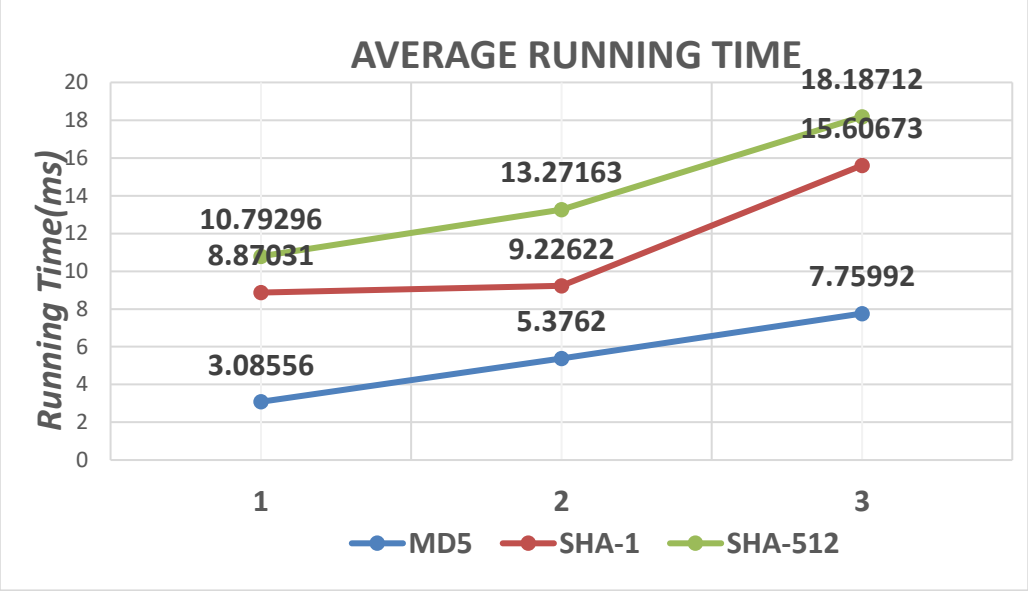


Figure 1. Chart about average running time MD5 and SHA-1 and SHA-512

The result shows that the running time of MD5 is faster than SHA-1, SHA-1 is faster than SHA-512.

*3.5. Comparison of the number of bytes encrypted per unit time*

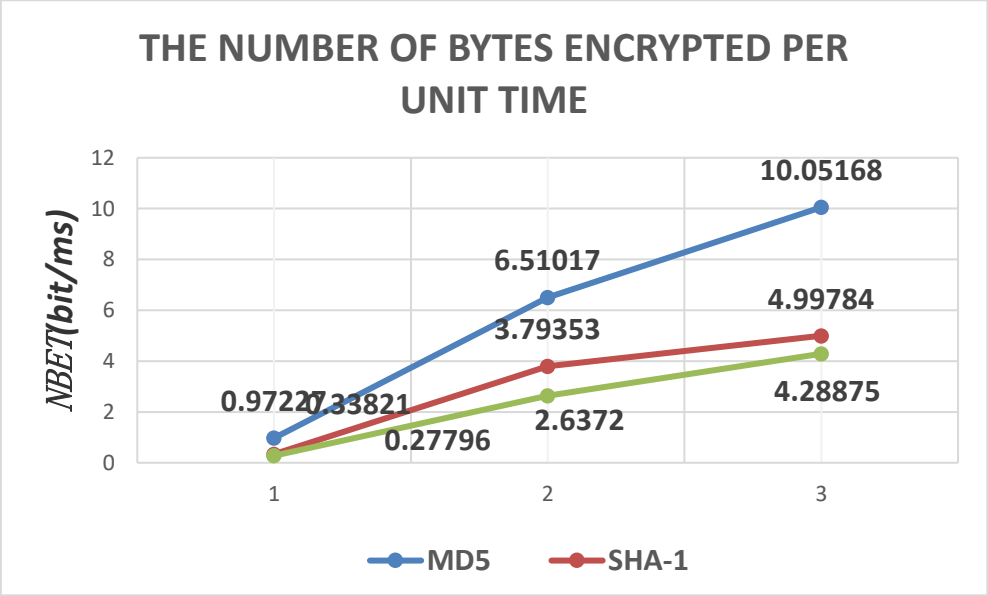


Figure 2. Chart about the number of bytes encrypted per unit time

It can be analyzed that when the number of encrypted strings is relatively small, the NBET of the three is similar, but when the number of strings increases, MD5 is more efficient than SHA-1, and SHA-1 is more efficient than SHA-512. In terms of decoding time, the order of the three algorithms is the opposite. However, according to the number of characters and encryption time of hash value, SHA-512 is better than SHA-1, and SHA-1 is better than MD5, if security is considered.

*3.6 Time Complexity of MD5 and SHA-1 and SHA-512*

MD5 Time Complexity

$$\sum = T(n) = \theta(n + n \times (16 + 16 \times 9 \times 2 + 16 \times 7 + 16 \times 8) + 4 + 3) = \theta(N)$$

SHA-1 Time Complexity

$$\sum = T(n) = \theta(n + n \times (16 + 64 \times 4 + 20 \times 15 + 20 \times 13 \times 2 + 20 \times 16) + 5 + 4) = \theta(N)$$

SHA-512 Time Complexity

$$\sum = T(n) = \theta(n + n \times (16 + 64 \times (2 \times 5 + 3) + 80 \times (6 + 14 + 11) + 8 + 7) = \theta(N)$$

It can be concluded that the time complexity of the three algorithms is the same. Finally, a table about three algorithms has been generated from the analysis above.

Table 1: Comparison of three algorithms

| Algorithm | Number of characters that output hash values | Time Complexity | Running Time(relatively) | NBET(relatively) | Block Size(bits) | Security(relatively) |
|---|---|---|---|---|---|---|
| **MD5** | 32 | $\theta(N)$ | Fast | High | 512 | Weak |
| **SHA-1** | 40 | $\theta(N)$ | Medium | Medium | 512 | Medium |
| **SHA-512** | 128 | $\theta(N)$ | Slow | Low | 1024 | Strong |

## 4. Suggestion

Users can choose these three encryption algorithms according to different requirements. When security is the priority, SHA-512 algorithm with the largest number of characters can be selected. When speed and efficiency of encryption is the first priority, MD5 algorithm can be selected. Considering speed and efficiency of encryption and security, SHA-1 algorithm can be chosen too.

## 5. Conclusion

Analyzing three hashing encryption algorithms generated the result of this study. Firstly, this article identified the time complexity of MD5, SHA-1, and SHA-512 is equal. Moreover, the running time of MD5 is faster than SHA-1, and SHA-1 is faster than SHA-512. Last but not least, the NBET OF MD5 is higher than SHA-1, and SHA-1 is higher than SHA-512 in most cases. Users can choose different encryption algorithms according to different requirements.

## References

[1] Coremen, T. H., Leiserson, C. E., et al. Introduction to algorithms, 2nd edition, MIT Press, 2009:254-277.
[2] Sedgewick, R., Algorithms in C, 3rd edition, Addison-Wesley, 1998:574-603.
[3] Zobel, J., Heinz, S., Williams, H. E., In-memory hash tables for accumulating text vocabularies. Information processing letters, 2001:271-277.
[4] Singh, M., Garg, D., Choosing best hashing strategies and hash functions[C], In 2009 IEEE International Advance Computing Conference, 2009:50-55.
[5] Rivest, R. The MD4 message digest algorithm[C], Advances in Cryptology-CRYPT0'90, 1991:303-311.
[6] Den Boer, B., Bosselaers, A., Collisions for the compression function of MD5[C], Workshop on the Theory & Application of of Cryptographic Techniques. 1993:293-304.
[7] Rachmawati, D., Tarigan, J. T., Ginting, A. B. C., A comparative study of Message Digest 5(MD5) and SHA256 algorithm, In Journal of Physics: Conference Series, vol.978, 2018.
[8] Lowekamp, B. B., Ed., US secure hash algorithm 1 (SHA1)[J]. 2010.
[9] Zhang, W. H., Tao, L. U., et al. A floristic study on seed plants in the upper reaches of Minjiang river [J]. Acta Botanica Boreali-occidentalia Sinica, vol.23, 2003:888-894.

[10] Gaj, K., Chodowiec, P., Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays, In Cryptographer's Track at the RSA Security Conference, 2001:84-99.
[11] Grembowski, T., Lien. R., et al. Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512[J]. 2002.
[12] Ahmad, I., Das, A. S., Analysis and detection of errors in implementation of SHA-512 algorithms on FPGAs [M]. Oxford University Press, 2007.