Hash Function Performance

Abner Adhiwijna 13516033[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516033@std.stei.itb.ac.id

***Abstract*—A hash function is a function that is used to map data of any sizes into fixed size data. A value from a hash function is created in such a way that it is difficult to find out the initial value. This is useful in storing data such as passwords so that the stored values cannot be used to infer the actual data. A brute force attempt to find a value similar to a certain value is possible. As the attempt requires to execute the hash function multiple times, the hash function's computing performance will have a big effect in the time it takes for a brute force attack. Thus, having a hash function that is significantly slower is more secure against a brute force attack. Various hash functions has different algorithms. This difference leads to functions having different performance that is important against a brute force attack.**

Keywords—hash; brute force; performance; computing time;

I. Introduction

A hash function maps data of arbitrary size to fixed size data. It works as a one-way function and designed to be infeasible to invert. Hash functions are widely used for cryptographic purposes for this reason. Password verification and digital signatures are applications of hash functions. Hash functions allows data such as passwords to be stored without revealing the password. However, this doesn't stop some people from trying.

A small change to a message being hash changes the hash value so significantly that it is near infeasible to find a specific message corresponding to a hash value. Brute force is the method used to crack hashes. The method simply tries all possible combinations until it arrives at a solution. If a hash function has low computing time, a brute force attack will be able to try more inputs in a period of time. Thus, it is important for hash functions to have some degree of performance to deter brute force attacks.

This paper will test several popular hash functions on an amount of inputs to check the comparative computing time of each of those functions. The time it takes to execute a function will help determine which hash function is more secure to use.

II. The Hash Functions Tested

A. Secure Hash Algorithm (SHA)

The Secure Hash Algorithms are cryptographic hash functions published by the Nasional Institute of Standards and Technology (NIST).

1) SHA-1

SHA-1 was first published at 1995. It produces a 160-bit hash value.

2) SHA-2

SHA-2 was published at 2001. It was designed by the US NSA. It has variants for 224, 256, 384, and 512 bits. This paper will cover SHA-256.

3) SHA-3

Also known as Keccak, this algorithm is fairly new first published at 2015. Keccak was selected as a winner of the NIST hash function competition for SHA-3.

B. MD Message-Digest Algorithm Series

This series was developed by Ronald Rivest.

1) MD4

MD4 was developed on 1990. It has now been replaced by MD5.

2) MD5

First published in 1992. Designed by Ronald Rivest to replace MD4

C. BLAKE2

BLAKE is a cryptographic hash function based on the ChaCha stream cipher. Blake was developed by Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan.

1) BLAKE2b

A variant of BLAKE2 optimized for 64-bit platforms.

2) BLAKE2s

A variant of BLAKE2 optimized for 8- to 32-bit platforms.

D. RipeMD

Developed in 1992 and has five variants which are RIPEMD, RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320. This paper will cover the most common one which is RIPEMD-160.

E. Whirlpool

A cryptographic hash function published at 2000 that is not patented and may be used freely. It was designed by Vincent Rijmen and Paulo S. L. M. Barreto.

III. The Environment

The test will be conducted against an array of random strings. Each string is randomized to have a length up to 100. For this test, Different amounts of string will be tried for all hash functions.

A. Hardware, Programming Language, and Libraries

The programming language that will be used for this test is python. The libraries used for this test is hashlib from python. Below are the list of functions used for each hash function.

- hashlib.sha1()
- hashlib.sha256()
- hashlib.sha3_256()
- hashlib.new('md4')
- hashlib.md5()
- hashlib.blake2b()
- hashlib.blake2s()
- hashlib.new('ripemd160')
- hashlib.new('whirlpool')

Below is the specification of hardware used for this test.

Operating System	Windows 10 Home Single Language 64-bit (10.0, Build 17134) (17134.rs4_release.180410-1804)
System Model	Inspiron 15 7000 Gaming
Processor	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz
Memory	16384MB RAM

B. String Values to be Hashed

Although this is insignificant for the performance of the hash function, it doesn't hurt to document how the random strings are generated. The maximum string length was set at 100. Each string will be generated from printable characters with a random length between 1 and the maximum length.

```
def initList(n=N):
    arr = []
    for i in range(n):
        arr.append(
            "".join(random.choice(
            string.printable)
            for _ in range(random.randint(
            0, maxRandLength)))
            .encode('utf-8'))

    return arr
```

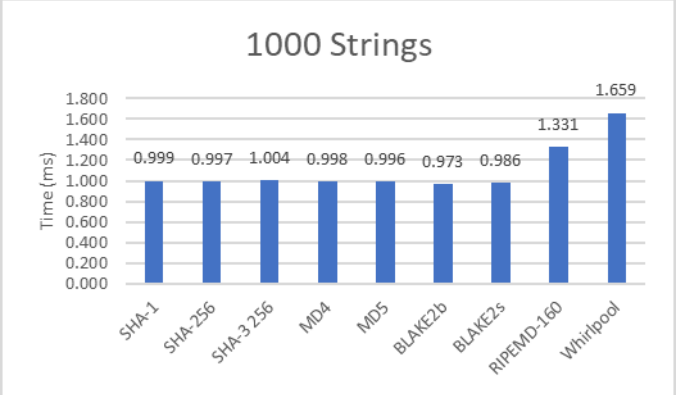
IV. Test Results

The test will be conducted three times for each amount of string. The value taken will be the average of those results. Time taken will be in milliseconds.

A. 1000 Strings

Function	Time 1 (ms)	Time 2 (ms)	Time 2 (ms)
SHA-1	1.00255	0.997066	0.997305
SHA-256	0.99659	0.996828	0.997782
SHA-3 256	0.99206	1.023293	0.997543
MD4	0.998974	0.997782	0.997543
MD5	0.994205	0.99802	0.99659
BLAKE2b	0.997543	0.92411	0.997543
BLAKE2s	0.997543	0.996351	0.962973
RIPEMD-160	0.997066	0.998735	1.996113
Whirlpool	1.98698	1.993895	0.997471

Function	Average Time (ms)
SHA-1	0.998974
SHA-256	0.997066
SHA-3 256	1.004299
MD4	0.9981
MD5	0.996272
BLAKE2b	0.973066
BLAKE2s	0.985622
RIPEMD-160	1.330638
Whirlpool	1.659449



Below is the speed sorted by time from fastest to slowest

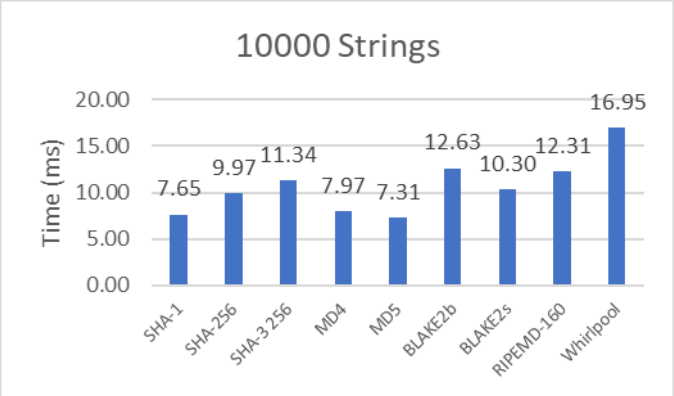
Function	Average Time (ms)
BLAKE2b	0.973066
BLAKE2s	0.985622
MD5	0.996272
SHA-256	0.997066
MD4	0.9981
SHA-1	0.998974
SHA-3 256	1.004299
RIPEMD-160	1.330638

Whirlpool	1.659449

B. 10000 Strings

Function	Time 1 (ms)	Time 2 (ms)	Time 2 (ms)
SHA-1	6.980658	7.950783	8.012056
SHA-256	11.95264	8.009434	9.939432
SHA-3 256	12.00318	10.99706	11.00588
MD4	6.949902	7.97987	8.988857
MD5	7.981777	6.981134	6.979227
BLAKE2b	13.98897	11.96814	11.94453
BLAKE2s	9.941816	9.004116	11.95264
RIPEMD-160	12.9993	12.96401	10.97488
Whirlpool	15.95616	18.93806	15.94949

Function	Average Time (ms)
SHA-1	7.647832
SHA-256	9.967168
SHA-3 256	11.33537
MD4	7.972876
MD5	7.314046
BLAKE2b	12.63388
BLAKE2s	10.29952
RIPEMD-160	12.31273
Whirlpool	16.94791



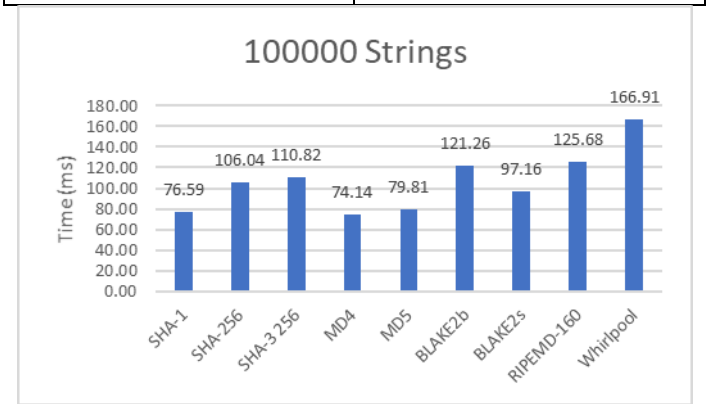
Below is the speed sorted by time from fastest to slowest

Function	Average Time (ms)
MD5	7.314046
SHA-1	7.647832
MD4	7.972876
SHA-256	9.967168
BLAKE2s	10.29952
SHA-3 256	11.33537
RIPEMD-160	12.31273
BLAKE2b	12.63388
Whirlpool	16.94791

C. 100000 Strings

Function	Time 1 (ms)	Time 2 (ms)	Time 2 (ms)
SHA-1	79.31113	74.77427	75.67954
SHA-256	110.6715	96.31968	111.1367
SHA-3 256	112.0458	112.7	107.7123
MD4	72.84069	78.78661	70.80388
MD5	72.83664	93.78076	72.80707
BLAKE2b	112.7012	119.7116	131.366
BLAKE2s	92.01121	103.7209	95.75677
RIPEMD-160	121.7146	122.6728	132.6454
Whirlpool	166.1785	168.5696	165.9915

Function	Average Time (ms)
SHA-1	76.58831
SHA-256	106.0426
SHA-3 256	110.8193
MD4	74.14373
MD5	79.80816
BLAKE2b	121.2596
BLAKE2s	97.16296
RIPEMD-160	125.6776
Whirlpool	166.9132



Below is the speed sorted by time from fastest to slowest

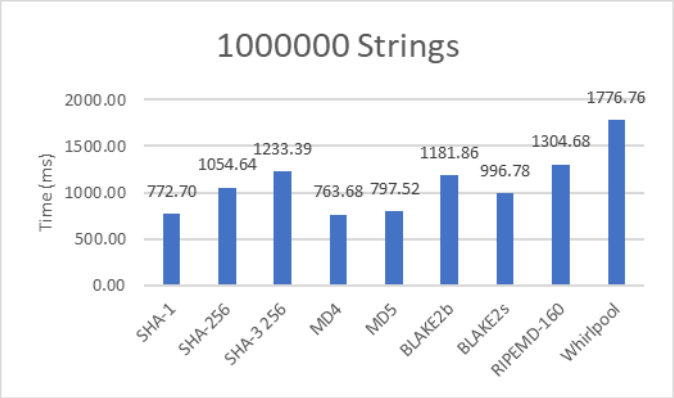
Function	Average Time (ms)
MD4	74.14373
SHA-1	76.58831
MD5	79.80816
BLAKE2s	97.16296
SHA-256	106.0426
SHA-3 256	110.8193
BLAKE2b	121.2596
RIPEMD-160	125.6776
Whirlpool	166.9132

D. 1000000 Strings

Function	Time 1 (ms)	Time 2 (ms)	Time 2 (ms)
SHA-1	759.9683	778.6429	779.4826

SHA-256	1000.324	960.1517	1203.443
SHA-3 256	1122.055	1111.153	1466.95
MD4	751.9937	731.5886	807.4577
MD5	762.4938	784.2755	845.7928
BLAKE2b	1149.026	1167.475	1229.07
BLAKE2s	964.6988	989.1622	1036.482
RIPEMD-160	1250.806	1294.768	1368.453
Whirlpool	1708.392	1796.22	1825.659

Function	Average Time (ms)
SHA-1	772.6979
SHA-256	1054.639
SHA-3 256	1233.386
MD4	763.68
MD5	797.5207
BLAKE2b	1181.857
BLAKE2s	996.7809
RIPEMD-160	1304.675
Whirlpool	1776.757



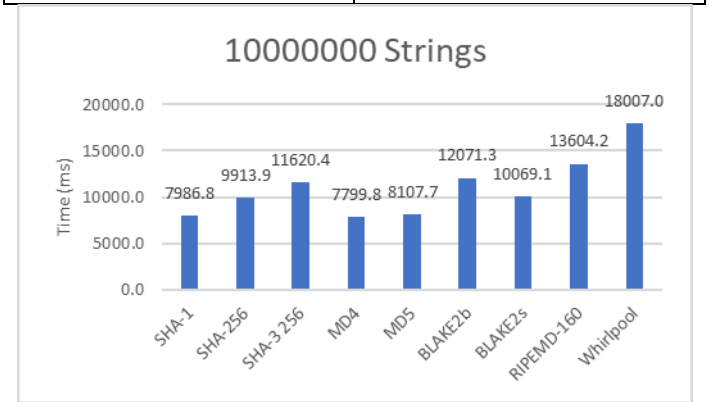
Below is the speed sorted by time from fastest to slowest

Function	Average Time (ms)
MD4	763.68
SHA-1	772.6979
MD5	797.5207
BLAKE2s	996.7809
SHA-256	1054.639
BLAKE2b	1181.857
SHA-3 256	1233.386
RIPEMD-160	1304.675
Whirlpool	1776.757

E. 10000000 Strings

Function	Time 1 (ms)	Time 2 (ms)	Time 2 (ms)
SHA-1	8137.61	7903.784	7918.985
SHA-256	9992.155	9966.959	9782.443
SHA-3 256	11462.22	12027.27	11371.63
MD4	7853.347	7882.942	7663.154
MD5	8162.451	8256.433	7904.304
BLAKE2b	11663.13	12524.95	12025.8
BLAKE2s	10165.77	10091.08	9950.375
RIPEMD-160	13447.09	13526.03	13839.56
Whirlpool	18226.76	18266.65	17527.65

Function	Average Time (ms)
SHA-1	7986.793
SHA-256	9913.852
SHA-3 256	11620.37
MD4	7799.814
MD5	8107.73
BLAKE2b	12071.29
BLAKE2s	10069.07
RIPEMD-160	13604.23
Whirlpool	18007.02



Below is the speed sorted by time from fastest to slowest

Function	Average Time (ms)
MD4	7799.814
SHA-1	7986.793
MD5	8107.73
SHA-256	9913.852
BLAKE2s	10069.07
SHA-3 256	11620.37
BLAKE2b	12071.29
RIPEMD-160	13604.23
Whirlpool	18007.02

V. Comparison and Analysis

A. Secure Hash Algorithm (SHA)

SHA-1, SHA-2, and SHA-3 has different durations for hashing the same amount of strings. There is a trend that SHA-2 generally is slower than SHA-2, and SHA-3 is slower than SHA-2. This means that over the course of new algorithms, the SHA series generally takes longer to hash a value. This is still a good change in terms of defending against brute force attacks

because it means that newer algorithms will need longer to be cracked.

B. *MD Message-Digest Algorithm Series*

MD5 generally takes a little bit more time than its predecessor, MD4. However, the difference is almost negligible. Hence, MD5 is not that much better in defending against brute force attacks in computing time than MD4

C. *BLAKE2*

BLAKE2s is faster then BLAKE2b. This is to be expected since BLAKE2b is optimized for 64-bit platforms while BLAKE2s is optimized to 8 to 32-bit platforms.

D. *General*

MD4, MD5, and SHA-1 have relatively fast computing times compared to the other hash functions. These functions are easy to attack and should be avoided.

SHA-256, SHA-3 256, BLAKE2b, and BLAKE2s have pretty good computing times. These hash functions should be safe to use just based on the security against brute force attacks.

> We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi resolution TIFF or EPS file with all fonts embedded) because this method is somewhat more stable than directly inserting a picture.
>
> To have non-visible rules on your frame, use the MSWord "Format" pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

RIPEMD-160 and Whirlpool both have a very high computing time. RIMEMD-160 needs about 36% more time than SHA-256 while Whirlpool is at 80% more. These functions are safe to use against brute force attacks.

VI. Conclusion

Every hash function has different performances and computing time. These computing times are useful for fending off brute force attacks. The more time it takes to execute a hash function, the more secure it is against brute force attacks.

Based on the findings, MD4, MD5, and SHA-1 has relatively low computing time and thus should not be used. These hash algorithms are susceptible against brute force attacks.

SHA-256, SHA-3 256, BLAKE2b, and BLAKE2s are safe to use with enough protection against brute force attacks.

RIPEMD-160 and Whirlpool are the slowest and should be the safest against brute force attacks. The functions are safe by increasing the amount of time needed for the same amount of executions.

Based on this test, it is concluded that RIPEMD and Whirlpool should be used. However, SHA-256, SHA-3, and BLAKE should not be forgotten when in need of more performance. MD4, MD5, and SHA-1 should not be used.

This paper does not cover other aspects of a hash function and it should be noted when choosing which function to use. Collision attacks are not covered here but are affected by the computing time needed by hash functions. Furthermore, newer algorithms has the advantage in that there has been less cryptoanalysis done to it. These aspects need to be considered outside of the computing time for fending off brute force attacks.

References

[1] https://web.archive.org/web/20130526224224/http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf Accessed on 09 May, 2019.

[2] https://keccak.team/, Accessed on 09 May, 2019.

[3] https://blake2.net/, Accessed on 09 May, 2019.

[4] https://homes.esat.kuleuven.be/~bosselae/ripemd160.html, Accessed on 09 May, 2019.

[5] https://web.archive.org/web/20171129084214/http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html, Accessed on 09 May, 2019.

[6] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[7] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[8] K. Elissa, "Title of paper if known," unpublished.

[9] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[10] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

Statement

I hereby declare that this paper is my own paper, not an adaptation, or a translation from someone else's paper, and not a plagiarism.

Bandung, 05 May 2019

Abner Adhiwijna 13516033