International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.79-83
ISSN: 2278-2397

Cryptographic Hash Algorithms Performance Finding using .Net Simulation

B.Nithya[*1], P.Sripriya[2]
[1]Research Scholar, Vels University, Pallavaram, Chennai.
[2]Associate Professor, Department of Computer Applications, Vels University, Pallavaram, Chennai.
Email: nithyababu.rch@gmail.com, sripriya.phd@gmail.com.

Abstract - Hashing is one of the cryptographic methods to provide security. It maps very big sets of keys to smaller sets of hash value. Hash algorithms convert any length of input to permanent length of output for fast accessing and to ensure integrity of data. Information authentication is provided by hash functions. Objective of this paper is to implement hash algorithms MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-160 in .Net platform and simulate the performances on the basis of speed, throughput and memory. The text files are taken for the tests to find the changes on these parameters when output length changes. The purpose is to get the performances are to take the best algorithm for information hiding.

Keywords**:** Hash, Secured Hash Algorithm, Message Digest, Integrity, Throughput.

I. INTRODUCTION

Hash algorithms are used in number of places currently in various security applications and in internet procedures. There are many hash algorithms in use but here the consideration is given to MD5 which is also known as Message Digest 5, SHA is abbreviated as Secured Hash Algorithm and RIPEMD (RACE Integrity Primitives Evaluation Message Digest). When checked[1] performances of MD5 and SHA with different parameters like key length, block size, rounds and total steps, SHA is safer than MD5. But MD5 is fastest than SHA. Only SHA family have been taken for performance check, and found that SHA-2 is secure and fit for wide-ranging use[2].

The platform MATLAB used to get results of SHA-256 and SHA-512 by applying avalanche effect. It is showed[3] that SHA-512 has highest time consumption than SHA-256. The research made on SHA-1 with android mobile verified better file integrity and much more enhancement to the file confidentiality[4]. Another study[5] was on SHA-1, SHA-192[1], SHA-192[2] to find avalanche effect, and proved that except SHA-192[1], others are indissoluble. Salt generation has been performed[6] using .Net and PHP to show the working status of hash algorithms. The authors[7] tried to hide the hash values behind the audio, video and image files. When visual studio .Net used for simulation, SHA-512 gives high-quality protection and SHA-1 consumes more time. Also there was the research of finding new hash algorithms like MD5-640[8, 9] bits. It grants high security when transferring data using 3G, 4G even on 5G. Evaluation has been prepared[9] for of SHA-1, SHA-256, RIPEMD-160, JERIM-320 using C complier to find speed and memory used for hashing original message. Where JERIM-320 produced much protected hash code than others because it has four parallel lines and makes differential attacks complicated.

II. CRYPTOGRAPHIC HASH FUNCTIONS AND SPECIFICATION

A cryptographic hash function H acquires an input of random piece and generates fixed length of message digest, which is usually known as H (M). This message digest has to convince some properties. The main properties are

i. Collision Resistance
ii. Preimage Resistance
iii. Second Preimage Resistance.

The first property is to prove that there should not be any two message digests are same is called “collision-freeness”. If there is H(M)=H(M’), the message digest is a collision resistance. The second is difficulty of finding message M from the hash value s. M1 should not be founded by M2 (H (M1) ≠H(M2)), if so that is second preimage resistance. If this property shows incorrect on any of the algorithm, this can be identified that the algorithm has been ‘broken’ or ‘attack’. Hash algorithms are applied[1] in MAC, Digital Signatures, Intrusion Detection, Virus Discovery and Pseudorandom Number Generator.

2.1 *MD5*

The message digest 5 algorithm produces 128 bit hash value as output for any length of input. It is used to find data integrity when downloading file from server, that is it cross checks the file whether the downloaded[10] file is same as the file which is actually stored in server. The steps for the MD5 algorithm,

i. Breaks up the input message to fixed size of 512 bits of blocks.
ii. If the last block is less than 512, extra bits are added to the end.
iii. Each block is divided into 16 words of 32 bits each denoted as M0, M1……M15.
iv. To process the 32 bits there is need of buffers, (a_i,b_i,c_i,d_i) where the initial value of $i=0$. That is called *initial value (IV).*
(a_0, $b0$,$c0$,$d0$) = (89452301_{16}, $efghdb89_{16}$, $98bsftfe_{16}$, 15697476_{16}).
v. And each four auxiliary function produces one 32 bit word from four 32bit input.
vi. After four rounds of 16 operations it provides the hash code.
$0123456989abcdfhbtcba9876544269_{16}$.

2.2 *SHA Family*

Secured Hash Algorithm includes,

i. SHA-0,
ii. SHA-1,
iii. SHA-2,
iv. SHA-3.

International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.79-83
ISSN: 2278-2397

Here in this paper SHA-1, SHA- 256, SHA-384, SHA-512 are picked and showed the differences by means of speed, throughput and memory. SHA-1 has same processing method as MD5, but it gives 160bit output length. The difference to SHAs are depends on their output length and the number of rounds it takes for the processing of input. Even a SHA-1 is a broken hash function[12], SHA-2 is a good one for security issues and currently SHA-2 is used by Wireless Local Area Networks and Firewalls. The processing steps are,

i. Padding bits – input size to 512bits
ii. Add length – at the end of padded input 64 bits are added
iii. The processing function includes ADD, OR and NOT
iv. Buffers to store the each final 32 bits
v. Processing input by 512 bits of blocks.

2.3 *RIPEMD-160*

With RIPEMD-160, there are also exist RIPEMD-128, RIPEMD-256 and RIPEMD-320. The RIPEMD- 160 is derived from MD4 algorithm and it separates the input message into blocks of 512bit. After that, 64 bit of original message will be added. Then 160 bit buffer is always ready to hold the immediate hash result.

III. EXPERIMENTAL ANALYSIS

The simulation has been undertaken on C#.Net and obtained the results of performances of hash algorithms. This brilliant language has predefined classes for the algorithms like MD5CryptoServiceProvider, SHA256, SHA384, SHA512, and the function ComputeHash() gives the hash code for the respective algorithms. The experiment has been taken place for text files from the size 877 kb to 21854kb. The performances of algorithms calculated in the basis of

a. Execution Time
b. Throughput
c. Memory Used

The execution time was calculated from the class StopWatch and its methods Start() and Stop(). The ComputeHash() method was given in between stopwatch.Start() and stopwatch.Stop() to find the time was taken to compute hash code. It tells the speed takes for process each algorithm. The throughput is to find how much bits are transferred per second. It is used to find the bandwidth of the network. The calculation of throughput was found by calculating the time divided by the input text file in kb. The memory used by each algorithm was estimated by finding the memory used before the ComputeHash() and after the memory used ComputeHash(). Finally subtracted memory after used from before memory used, to get the actual space occupied for the process.

IV. RESULT AND DISCUSSION

Authors of previous work on hash algorithms are based on keys, length, block size, rounds, steps taken by them, internal state size, avalanche effect, collision effect, hash message size. With these parameters, comparisons made and gave the results. But less research made on finding speed of the algorithm, memory used and the throughput of the hash algorithm. This work takes these parameters to find the performances. Also here various sizes of text file inputs are used to find the performance. The text files from 877kb to 21854kb are given as input to each algorithm one by one. The simulation gives the results for speed, throughput and memory used. The execution time was given as chart (Figure 1). From this SHA-512 and SHA-384 are acquired highest execution time (Table.2). The SHA-1 takes lowest time to produce hash code even for the biggest text file. The RIPEMD also consumes least time. SHA-256 and MD5 speed is moreover identical.

Table 1. Previous Work with the Cryptographic Hash Algorithms

Authors	Algorithms	Parameters	Previous Results
Piyush Gupta[1] et.al	MD5,SHA	• Key • Length • Block Size Cryptanalysis • Rounds • Total Steps	SHA is safer than MD5 but MD5 is fastest than SHA on 32 bit machines
B. Madhuravani [2] et.al	MD5, SHA-0, SHA-1, SHA-2, SHA-3.	• Output size • Internal State Size • Block Size • Message Size • Rounds	Performance of SHA-2 in hardware implementation is good and it is secure than other SHA family
Smriti Gupta[3] et.al	MD2,MD4,MD6,SHA160,SHA256,SHA512	• Avalanche Effect • CPU Cycle • K-S Test • Chi-Square	SHAs are safest than MDs, but MDs have time consumption
Hanumantu[4] et.al	MD5,SHA-1, SHA-512	• Android Mobile File integrity check	SHA-1 showed best performance
Kamlesh kumar Raghuvanshi [5]et.al	MD5, SHA-0, SHA-1, SHA-2, SHA-192[2], SHA-192[1]	• Rounds • Output Size • Execution time • Collision found • File sizes upto 10kb	SHA-2 has highest execution time but proved unbroken

		• Avalanche Effect	
Pritesh N. Patel[6] et.al	SHA-256	• Salt Generation using PHP and .Net	Salt generation prevents attacker from hacking of hash codes.
Sangeeta Raheja[7] et.al	SHA-1, SH256, SHA512, LSB	• Rounds • Output Size • Execution time • Collision found • Message size • Used .Net simulation	SHA-512 is best
Deepika Sharma[8] et.al	New MD5-640	• Server side- Client side implementation at Ubuntu	Improved security than regular MD5
Sheena Mathew[9] et al.	SHA-1 SHA-256 RIPEMD-160 RIPEMD-320 JERIM-320	• Rounds • Output Size • Execution time • Collision found • Message size • Speed • Memory	JERIM-320 is more secure than other hash because it makes differential attacks complex.

Table 2. Execution Time for various hash algorithms in Ms

Text files (KB)	MD5(128)	RIPEMD(160)	SHA1	SHA256	SHA384	SHA512
877	5.181	43.169	13.646	55.478	83.386	153.072
1653	20.703	80.3893	23.913	82.523	193.794	148.347
3758	21.231	96.935	67.958	187.98	330.43	333.079
5071	85.704	130.4847	41.488	229.214	429.757	428.764
10021	190.69	247.774	63.46	436.442	860.639	827.1413
15034	758.68	369.1593	101.0388	647.784	1240.463	1240.36
21854	385.93	525.607	322.555	932.596	1809.111	1817.312

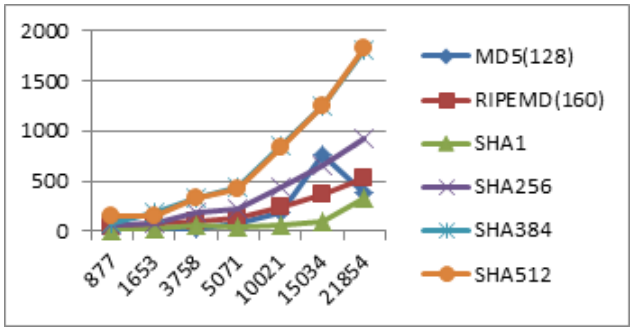

Figure 1. Execution Time

Table 3. Throughput of various hash algorithms (Bits/S)

Text files (KB)	MD5(128)	RIPEMD(160)	SHA1	SHA256	SHA384	SHA512
877	6.054	80.438	15.944	64.81	97.426	178.84
1653	12.489	49.804	14.815	51.126	120.063	91.906
3758	5.724	26.137	18.324	50.689	89.09	89.81
5071	17.308	26.351	8.378	46.29	86.79	86.59
10021	19.487	25.319	6.484	44.598	87.94	84.523
15034	51.677	25.145	6.88	44.123	84.494	84.48
21854	18.083	24.628	15.11	43.698	84.768	85.153

International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.79-83
ISSN: 2278-2397

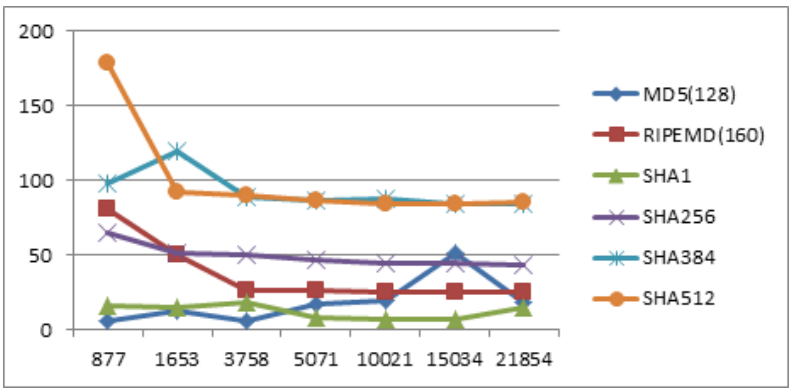

Figure 2: Throughput

Table 4: Memory Used by various hash algorithms (Kb)

Text files (KB)	MD5(128)	RIPEMD(160)	SHA1	SHA256	SHA384	SHA512
877	184	192	180	184	188	192
1653	184	192	180	184	204	192
3758	184	192	176	184	188	192
5071	184	192	180	184	188	192
10021	184	192	180	184	188	192
15034	184	192	180	184	188	192
21854	184	192	180	184	188	192

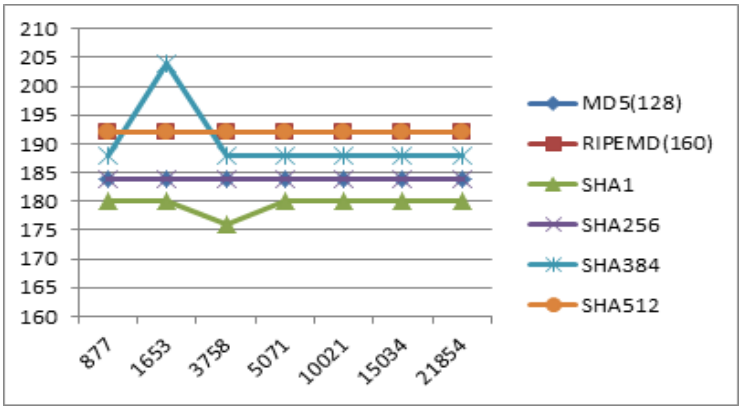

Figure 3: Memory Used

Throughput is a main metric of an algorithm's performance discovering. It is a calculation that bits/s transferred through a network and the bandwidth capacity of the network. The throughput chart (Figure.2) shows that SHA-512, SHA-384 have highest throughput. But MD5 and SHA-1 are illustrated as low throughput. SHA-512 and SHA-384 have the biggest length of output but these are having good throughput. MD5 and SHA1 have lowest output length also having least throughput. Here the files sizes are not concerned. Because all the algorithms have the same range of bits transferred when the file sizes are changed except RIPEMD (Table 3).

The performance of an algorithm is also concerned with the space that the algorithm takes for the process. The memory used for hash algorithms are calculated and proved that whatever may be the file size, the algorithms SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD, MD5 used equal space to compute hash (Figure 3). If 180 kb has been taken for 877kb file, the identical space has been used for the file size 21854. Here MD5 used less space when match up to other algorithm and SHA 512 takes high space for the process (Table 4).

V. CONCLUSION

Implementation of hashing algorithms SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD, MD5 are done on .Net platform. And the results showed that SHA-384 and SHA-512 are greatest in throughput. These both have highest hash size and gives better performance on throughput. But the parameters speed and memory present a finest report to MD5 than others. This research proved that the size of output makes the differences to parameters of speed, throughput and memory not because of the file size. Also biggest hash code producing algorithm shows a best performance. The Future work will be on SHA-384 and SHA-512 to find the attacks of hash

International Journal of Computing Algorithm
Volume: 05 Issue: 02 December 2016 Page No.79-83
ISSN: 2278-2397

algorithm and to strengthen these two for providing better security.

References

[1] Piyush Gupta, Sandeep Kumar, "A Comparative Analysis of SHA and MD5Algorithm" International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, 4492-4495.
[2] B. Madhuravani, D. S. R Murthy, "Cryptographic Hash Functions: SHA Family", International Journal of Innovative Technology and Exploring Engineering Vol. 2(4), 2013, 326-329.
[3] Smriti Gupta, Sandeep Kumar Yadav, "Performance Analysis of Cryptographic Hash Functions", International Journal of Science and Research, Vol.4 (8), 2015, 864-867.
[4] Hanumantu Rajeswari , Ramesh Yegireddi , Vudumula Govinda Rao, "Performance Analysis of Hash Algorithms and File Integrity " International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 7376-7379.
[5] Kamlesh kumar, Raghuvanshi Purnima, Khurana Purnima Bindal, "Study and Comparative Analysis of Different Hash Algorithm", Journal of Engineering Computers & Applied Sciences, Vol. 3 (9), 2014, 1-3.
[6] Pritesh N. Patel, Jigisha K. Patel and Paresh V. Virparia, "A Cryptography Application using Salt Hash Technique", International Journal of Application or Innovation in Engineering & Management, Vol. 2 (6), 2013, 236-239.
[7] Sangeeta Raheja Shradha Verma, " Comparative study of Hashing Algorithm Using Cryptographic and Steganography Using Audio Files", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4(5), 2014, 292-295.
[8] Deepika Sharma, Pushpender Sarao," Implementation of Md5- 640 Bits Algorithm", International Journal of Advance Research in Computer Science and Management Studies, Vol. 3(5), 2015, 286-293.
[9] Sheena Mathew, K. Poulose Jacob, "Performance Evaluation of Popular Hash Functions", World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol 4(1), 2010, 65-68.
[10] Nithya B, Sripriya.P. A Review of Cryptographic Algorithms in Network Security, International Journal of Engineering and Technology, 2016, 8(1), pp.324-331.
[11] William Stallings, A Book: Cryptography and Network Security Principles and Practice, Fifth Edition, 2006.
[12] Marc Stevens, A Book: Attacks on Hash Functions and Applications, 2000.
[13] Mohammad A. Alahmad, Imad Fakhi Alshaihli, "Broad View of Cryptographic Hash Functions", International Journal of Computer Science Issues, Vol 4(1), 2013.