International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Performance Analysis of Cryptographic Hash Functions

Smriti Gupta[1], Sandeep Kumar Yadav[2]

[1]M.Tech Student ,Government Mahila Engineering College, Ajmer, India

[2]Assistant Professor, Government Mahila Engineering College, Ajmer, India

Abstract: ***This paper presents the design and analysis of cryptographic hash functions.A hash function is a map from variable-length input bit strings to fixed-length output bit strings. Despite their simple definition, hash functions play an essential role in a wide area of applications such as digital signature algorithms, message authentication codes, password verification, and key derivation. The main contribution of this paper is to obtain some important parameters for a cryptographic hash function for comparative study. In this paper, we approach the problem of the design and analysis of cryptographic hash functions. We cover the design aspects of some hash functions in this paper and then we are analyzing the result of cryptographic hash functions.***

Keywords: Message Digest(MD'S), Secure Hash Algorithm(SHA), Strict Avalanche Criteria(SAC), Kolmogorov Smirnov Test , Chi Square Test.

1. Introduction

A cryptographic hash function H is a map from variable-length input bit strings to fixed-length output bit strings, H: {0, 1}∗ → {0, 1}n .On the other hand, a cryptographic hash function can be defined more formally as an instance from a family of functions. Let H :{0, 1}k × {0, 1}∗ → {0, 1}n be a family of functions. For a particular key K ∈ {0, 1}k, HK : {0, 1}∗ → {0, 1}n is defined for every m ∈ {0, 1}∗ by $HK(m) = H(K,m)$. In practice when we refer to a hash function we mean this instance. If the key K is secret then the hash function is used for authentication and it is called a message authentication code abbreviated as MAC. To simplify the notations we will drop K most of the time. Hash functions compress the input that is the domain of the input is larger than the range, hence collisions are unavoidable. However, a secure hash function should be collision-resistant, meaning that it should be hard to find collisions. But a collision can be found accidentally or computed in advance; to overcome this problem in formal proofs one has to find a collision for each member of the family, which makes it harder to pre-compute the collision for each key. Hence, in order to define formal security notions, cryptographic hash functions are defined as families. Namely, in [1] Damgard introduces infinite family of hash functions that captures his definition of computational infeasibility. But this method is not applicable to practical concrete constructions, because it gives asymptotic results.

Hence finite families of hash functions are used in formal security proofs. Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value. A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. As a minimum, it must have the following properties [2]:

(A) Preimage resistance
Given a hash h it should be difficult to find any message m such that $h = hash(m)$. This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.

(B) Second-preimage resistance
Given an input m_1 it should be difficult to find another input m_2 where $m_1 \neq m_2$ such that $hash(m1) = hash(m2)$. This property is sometimes referred to as weak collision resistance, and functions that lack this property are vulnerable to second-preimage attacks.

(C) Collision resistance
It should be difficult to find two different messages$m_1$and $m_2$such that $hash(m_1) = hash(m_2)$. Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for preimage-resistance, otherwise collisions may be found by a birthday attack.

These properties imply that a malicious adversary cannot replace or modify the input data without changing its digest. Thus, if two strings have the same digest, one can be very confident that they are identical.

A function meeting these criteria may still have undesirable properties. Currently popular cryptographic hash functions are vulnerable to length-extension attacks: given $h(m)$ and $len(m)$ but not m, by choosing a suitable m'an attacker can calculate $h(m||m')$ where || denotes concatenation. This property can be used to break naive authentication schemes based on hash functions. The HMAC construction works around these problems.

Ideally, one may wish for even stronger conditions. It should be impossible for an adversary to find two messages with substantially similar digests; or to infer any useful information about the data, given only its digest. Therefore, a cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable.

2. Characteristics and Assessment of Performance

MD-2:- MD-2 takes a message equal to an arbitrary number of 8-bit bytes and produces a 128 bit message digest. It cannot handle a message that is not an integral number of bytes, though it would be simple to modify MD-2, or to have a convention for bit padding message before feeding it to MD-2 [5]. The basic idea behind MD-2 is as follows:

- The input to MD-2 is a message whose length is an arbitrary number of bytes.
- The message is padded according to specified conventions, to be a multiple of 16 bytes.
- A 16 byte quantity, which MD-2 calls a checksum, is appended to the end. This checksum is a strange function of the padded message defined specifically for MD-2.
- Final pass- The message is processed, 16 bytes at a time, each time producing an intermediate result for the message digest. Each intermediate value of the message digest depends on the previous intermediate value and the value of the 16 bytes of the message being processed.

MD-4:- MD-4 was designed to be 32 bit word oriented. MD-4 can handle message with an arbitrary number of bits. Like MD-2 it can be computed in a single pass, though MD-4 needs more intermediate states [5].

- In MD-4 message digest to be computed is a 128 bit quantity (four 32 bit words). The message is processed in 512 bits (sixteen 32 bit words) blocks. The message digest is initialized to a fixed value, and then each stage of the message digest computation takes the current value of the message digest and modifies it using the next block of the message. The final result is the message digest for the entire message.
- Each stage makes 3 passes over the message block. Each block has a slightly different method of mangling the message digest. At the end of the stage, each word of the mangled message digest is added to its pre-stage value (which becomes the pre-stage value for the next stage). Therefore, the current value of the message digest must be saved at the beginning of the stage so that it can be added in at the end of the stage. Each stage starts with a 16 word message block and a 4 word message digest value [6].

MD-5:-MD-5 was designed to be somewhat more 'conservative' than MD-4 in terms of being less concerned with speed and more concerned with security. It is very similar to MD-4. The major differences are:

- MD-4 takes 3 passes over each 16 byte chunk of the message. MD-5 makes 4 passes over each 16 byte chunk.
- The functions are slightly different, as are the number of bits in the shifts.
- MD-4 has one constant which is used for each message word in pass 2, and a different constant used for the entire 16 message words in pass 3. No constant is used in pass 1.

MD-5 uses a different constant for each message word on each pass. Since there are 4 passes, each of which deals with 16 message words, there are 64 32-bit word constants used in MD-5.

The message digest in MD-5 is a 128 bit quantity (four 32-bit words). Each stage consists of computing a function based on the 512 bit message chunk and the message digest to produce a new intermediate value for the message digest. The value of the message digest is the result of the output of the final block of message.

Each stage in MD-5 takes four passes over the message block. At the end of the stage, each word of the modified message digest is added to the corresponding pre-stage message digest value [5].

SHA: This Standard specifies five secure hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. All five of the algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation also called as a message digest.

Each algorithm can be described in two stages**:** preprocessing and hash computation. Preprocessing involves padding a message, parsing the padded message into m-bit blocks, and setting initialization values to be used in the hash computation. The hash computation generates a message schedule from the padded message and uses that schedule, along with functions, constants, and word operations to iteratively generate a series of hash values. The final hash value generated by the hash computation is used to determine the message digest. The five algorithms differ most significantly in the security strengths that are provided for the data being hashed. Here, only 3 of them SHA-1, SHA-256 and SHA-512 described in detail because SHA-224 and SHA-384 are almost same as SHA-256 and SHA-512 respectively [8].

3. Case Study and Results

(A) Tools Used in Simulation

For the simulation of the described work, laptop with core-i5 64-bit microprocessor at 2.4 GHz, having 4GB RAM is used as machine, while the MATLAB 7.8 launched in February 2009, as a 64-bit software is employed. For the compilation of the report, Microsoft Office 2007 is used with their tools like Equation Editor, Visio and Picture Manager.

(B) Results

The conventional and proposed work are correctly simulated and output of the conventional Message Digests has been cross checked with published example in FIPS-180-3 [5]. Among several available parameters regarding the performance, a few are taken into account and analyzed after the simulation. These parameters are Message Digest calculation time, number of CPU cycles consumed. As the security parameters, some of the randomness tests have been performed to check the results with strict avalanche criteria.

(1) Hash Calculation Time

The message digest calculation time is one of the very important parameter while observing performance of any algorithm. The observed time is in seconds.

Table 1: Time consumed by various hash algorithms

Hash Algorithms	MD-2	MD-4	MD-5	SHA160	SHA-256	SHA-512
Time consumed	0.016	0.016	0.016	0.0251	0.031	0.4

Figure 1: shows the time to calculate message digest of one block.

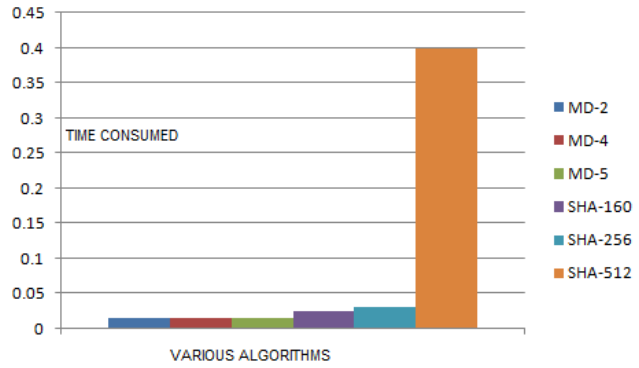

Figure 1: Time consumed by various hash algorithms

(2) CPU Cycles Consumed

In order to examine the hardware efficiency of the system for the particular algorithm, we must watch through its cycles during message digest calculation. So, here also another performance parameter is taken into account and plotted in Fig.2.

Table 2: Cycles consumed by various hash algorithms

Hash Algorithms	MD-2	MD-4	MD-5	SHA-160	SHA-256	SHA-512
CPU cycles consumed ($\chi 10^5$)	0.178	0.256	0.317	0.665	0.7301	3.032

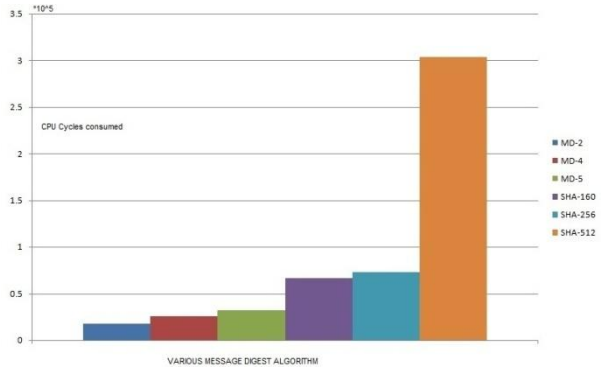

Figure 2: Cycles consumed by various hash algorithms

(3) Strict Avalanche Criteria

The analyses of security of various systems, taken under observations, were important, as we must know that all the system whether falls under an optimum level of security or not. To account the security, two parameters are taken: SAC and BIC. The strict avalanche criterion (SAC) is a generalization of the avalanche effect. It is satisfied if, whenever a single input bit is complemented, each of the output bits changes with a 50% probability [2]. The SAC builds on the concepts of completeness and avalanche. The outcome of the test is in terms of probability and plotted here in Fig.3.

$$K_{SAC}(i,j) = \frac{1}{2^n}\left(a_j^{ei}\right) = \frac{1}{2}$$

Where, $K_{SAC}(i,j)$ can take values in the range [0,1], and it should be interpreted as the probability of change of the j^{th} output bit when the i^{th} bit in the input string is changed and, $W(a_j^{ei})$ is input word to the system.

Table 3: Avalanche coefficient for various hash algorithms

Hash Algorithms	MD-2	MD-4	MD-5	SHA-160	SHA-256	SHA-512
Avalanche Coefficient	0.593	0.484	0.422	0.4875	0.5391	0.5098

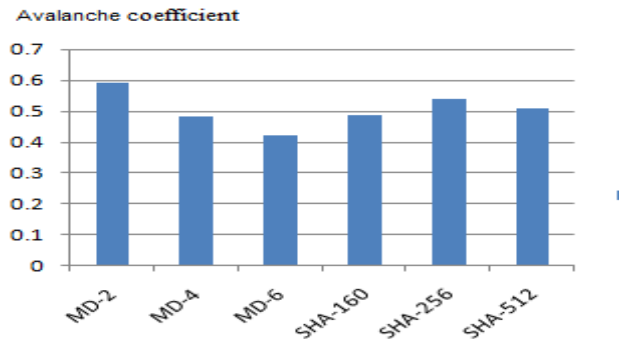

Figure 3: Avalanche coefficient for various hash algorithms

(4) Kolmogorov SMIRNOV Test

The K-S test is based on largest absolute deviation between uniformly distributed continuous CDF and empirical CDF. In the process the input data is ranked in ascending order and maximum values are observed in particular interval [3].

$$D^+ = \max_{1 \le i \le N}\left\{\frac{i}{N} - R(i)\right\}$$

$$D^- = \max_{1 \le i \le N}\left\{R(i) - \frac{i-1}{N}\right\}$$

$$D = max\,(D^+, D^-)$$

Where $R(i)$ is ranked input data, N is total number of data and i is the index.

Table.4: K-S test values for various hash algorithms

Hash Algorithms	MD-2	MD-4	MD-5	SHA-160	SHA-256	SHA-512
Threshold value	0.392	0.392	0.392	0.356	0.2881	0.2037
Obtained value	0.187	0.125	0.25	0.3102	0.1211	0.1055

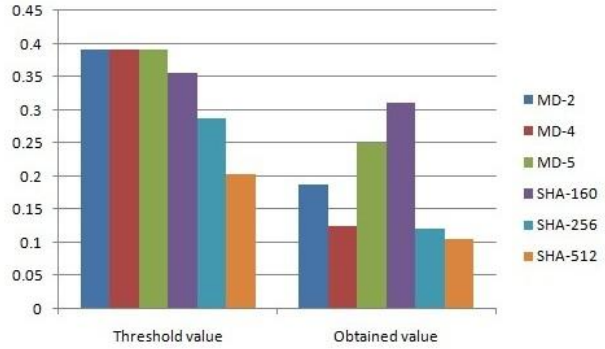

Figure 4: K-S test

Obtained maximum value is compared with critical value and hypothesis is rejected if obtained value is greater than critical value. The observed output is plotted in figure.4.

(5) CHI SQUARE TEST

It uses the sample statistic as

$$X_0^2 = \sum_{i=1}^{n} \left(\frac{(O(i) - E(i))^2}{E(i)}\right)$$

Where $O(i)$ is the observed number in i^{th} class, $E(i)$ is expected number in i^{th} class and n is the number of classes. The expected number is given by $E(i) = N/n$; where N is total number of observations. The outcome of test is compared with critical value to satisfy the hypothesis [3]. The observed output is plotted in figure.5.

Table 5: Chi-Square test values for various hash algorithms

Hash Algorithms	MD-2	MD-4	MD-5	SHA-160	SHA-256	SHA-512
Threshold value	16.9	16.9	16.9	16.9	16.9	16.9
Obtained value	5.25	7.75	16.5	8.0	7.375	12.875

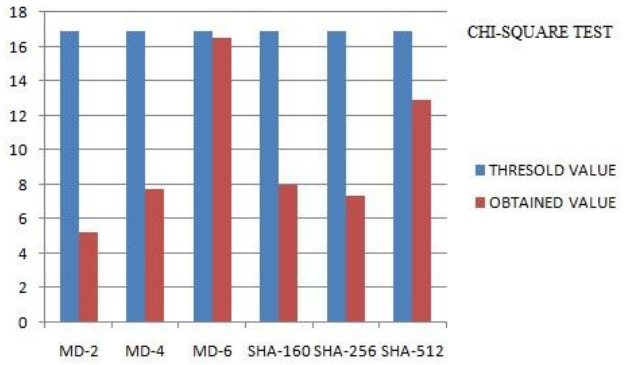

Figure 5: Chi-square test

4. Conclusion

Cryptographic hash functions are very important in current communication scenario. Many algorithms have been used to generate message digest or message authentication codes. As the computational speed and attacks are increasing day by day MD's are no longer called as secure and efficient message digest algorithm. SHA's are more secure algorithms than MD's. Although only SHA-160 found under attack theoretically not practically, It does not possess that level of security but still is very popular. SHA-256 and SHA-512 are safe and secure till date.

So SHA-512 is more secure algorithm than the MD-2, MD-4, MD-5,SHA-160 and SHA-256 ,But there is trade off between security and performance of SHA-512 algorithm. Time consumption to evaluate hash is high in SHA-512.But security is the main concern in most cases but this drawback can be overcome by a proper combination of MD's and SHA -512 which will give the optimized Hash algorithm regarding speed and security

References

[1] I Damgard. Collision free hash functions and public key signatureschemes. In D. Chaum and W. L. Price, editors, Advances in Cryptology-EUROCRYPT '87, Workshop on the Theory and Application of Cryptographic Techniques, Amsterdam, The Netherlands, April 13-15, 1987,Proceedings, volume 304 of Lecture Notes in Computer Science, pages 203–216. Springer, 1988.

[2] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," EUROCRYPT, pp. 1-18, 2005.

[3] Isel V., and Melek D. U, Avalanche and Bit Independence Properties for Ensemble of Randomly Chosen nXn S-Boxes, Tubitak Vol. 9, No. 2, 2001.

[4] Banks J., Carson J. S. II, Nelson B. L., and Nicol D. M., Discrete-event System Simulation., Delhi: Pearson Education PTE Ltd., 2005.

[5] Kaufman C., Perlman R., Speciner M., Network Security: Private Communication in Public World, New Delhi: McGraw Hill, 2002

[6] Boer B., Bosselaers A., "An Attack on the Last Two Rounds of MD4," ACM digital library, 1991.

[7] Stalling W., Cryptography and Network Security. New Delhi: McGraw Hill, 2002.

[8] Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions," EUROCRYPT, pp. 19-35, 2005.