Cryptographic Hash Functions: SHA Family

B. Madhuravani, D. S. R Murthy

Abstract: This paper gives the Study of Cryptographic Hash functions, which plays a vital role in the Security of many applications such as Digital Signatures, Tamper Detection, Password Protection and so on. We start with reviewing basic fundamentals of hashing. Hash functions are being used as building blocks of many complex Cryptographic mechanisms and protocols. Most of the widespread and popular hash functions, such as MD5, SHA-1 and SHA-2 share a common design philosophy. Recent Cryptanalytic advances have raised serious concerns regarding the long–term security of these hash functions. Security flaws have been detected in some of the most commonly used hash functions like MD5 and SHA-1. Even though the SHA-2 family is not really threatened by any attack, it receives little confidence because it is based on the same design principles. The dedicated hash functions from SHA family – SHA0, SHA-1, SHA-2, SHA-3 are compared in this paper.

Keywords: Hash Function, MD5, SHA-0, SHA-1, SHA-2, SHA-3.

I. INTRODUCTION

A hash function maps a variable-length input into a fixed-length output. This hash function output can be treated as a *fingerprint* of the input data [1]. Cryptographic hash functions have an important role in many applications. Digital signature, message integrity, authentication protocols, password protection and random number generation are some of them. Digital Signature Scheme (DSS) is used for authentication of data. Hash function is also used in authentication protocols such as Kerberos. Kerberos offers authentication, eavesdropping prevention, and integrity of data in client-server architecture. Kerberos uses hash function to calculate the hash value of the given client password and this hash value becomes the secret key of the client. Secure communication protocols such as Internet Protocol Security (IPSec), Secure Socket Layer (SSL), or Secure Shell (SSH) also use hash functions. The handshaking protocol in SSL uses a hash function to create a message authentication code. Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extensions (S/MIME) also use hash function to ensure the integrity of e-mail messages.

Manuscript received on March, 2013.

B. Madhuravani, Assistant Professor, Department of CSE, MLR Institute of Technology, Dundigal,Hyderabad. India.

Dr. D. S. R. Murthy, Professor in Information Technology, SNIST, Hyderabad. India.

Therefore, any flaw in a hash algorithm affects various applications. For several years standard Secure Hash Algorithm–1 **(**SHA−1), SHA−2 and non-standard (Message Digest–5) MD−5 hashing algorithms are widely used. However, cryptanalytic attacks, especially attacks of Wang et al. [2, 3, 4, 5], and increasing computational power brings the need for a new and more secure hash function. For this reason, National Institute of Standards and Technology (NIST) announced a publicly available contest, similar to the Advanced Encryption Standard (AES) contest, in which the winning algorithm keccak is selected as the new standard hashing algorithm SHA−3.

The paper is organized as, the basic properties of a cryptographic hash functions have been introduced in Section II. Section III explains the construction methods of hash function. Section IV gives descriptions of hash functions in the SHA family: SHA-0, SHA-1, SHA-2, SHA-3 and its comparison in Section V.

II. PROPERTIES OF CRYPTOGRAPHIC HASH FUNCTION

The three main properties which a “good” cryptographic hash function has to preserve [6].

1. *Preimage Resistance*: Given a digest y, it is computationally infeasible to find a message x that hashes to y. That is, computational cost of finding the input x must be $\geq 2^n$, where h(x) = y and |y| = n.
2. *Second Preimage Resistance*: Given a message x, it is computationally infeasible to find a different message x', such that both messages hash to a same digest. That is, computational cost of finding the input x' ($\neq$ x) must be $\geq 2^n$, where h(x') = y, h(x) = y, and |y|=n.
3. *Collision Resistance*: It is computationally infeasible to find two different messages, which hash to the same digest. That is, computational cost of finding an input pair x and x' such that h(x) = h(x') must be $\geq 2^{n/2}$. Here n is the length of message digest.

The preimage resistance property can be expressed as the inability to learn about the contents of the input data from its digest. The second preimage resistance property can be interpreted as the inability to learn about the second preimage from the given first preimage such that both of these preimages have same digest. The collision resistance property signifies that the digests are almost unique for each given message. These three properties - preimage resistance, second preimage resistance, and collision resistance are also known as one-way, weak collision resistance, and strong collision resistance properties respectively. If a hash function satisfies the first two properties then it is referred as *One–Way Hash Function* (OWHF). Whereas the hash function that satisfies all the three properties referred as *Collision Resistant Hash Function* (CRHF) [7].

III. CONSTRUCTION METHODS

Construction methods of hash function include Merkle-Damgard (MD) 1, HAsh Iterative FrAmework (HAIFA) and some other alternative designs derived from Merkle-Damgard Sponge construction is a new and secure construction method which is becoming more popular.

A. Merkle-Damgard (MD)

Merkle-Damgard construction is the most widely used hash construction method, which was designed by R. Merkle [8] and I. Damgard [9] independently in 1989. Most of the hash functions and all the standardized hash functions are build upon MD construction.

MD construction (Fig. 1) is basically processed in three steps. First step is the padding step. The aim of the padding is to make the message length a multiple of message block length, m. The most widely used padding procedure is as follows: a '1' bit followed by a number of '0' bits and the bitwise notation of the message length are appended to the message. The number of '0' bits appended to the message are chosen so that the length of the message becomes a multiple of block length m. In general the maximum length of the message that can be processed by the hash function is $2^{64} - 1$. Therefore 64 bit space is provided for the length padding.

Considering the appended '1' bit, at least 65 bits are appended to all messages regardless of the message length. If *l* is the length of the message, d, the number of '0' bits appended is the smallest positive root of the equation

$l + 65 + d \equiv 0 \pmod{m}$.

Second step is dividing the padded message into m bit blocks m0m1m2 . . . m_{t-1}. After this step, the chaining values are iteratively found by using a fixed, publicly known initialization vector, IV, and the message blocks:

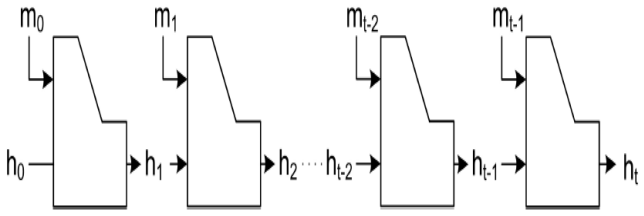
$h_0 = IV$

$h_i = f(h_{i-1}, m_{i-1})$ $i = 1, 2 \ldots, t$

where f is the compression function of the hashing algorithm.

Generally ht is taken to be the hash value of the message. However an optional transformation g(x) can be applied to ht to get the hash value as

$H(M) = g(h_t)$.



FFig1. MD Structure

The iterative structure of MD construction enables arbitrary length messages to be processed easily by the hash functions. Also padding the message length and using a non-zero IV, which is called is called the MD−strengthening, increases the defeats or increases the complexity of various attacks.

The most important property of MD structure is that the collision resistance property of the compression function is preserved [8, 9]. The MD structure has a security proof that states if the compression function used in the hash algorithm is collision resistant then the hash function itself is collision resistant.

Besides collision resistancy, it was believed that MD construction also preserves the pre image resistancy and second pre image resistancy of the compression function [10]. However there are several attacks in recent years against the second pre image resistancy of the MD construction such as [11].

B. HAIFA

The HAIFA designed by Biham and Dunkelman, is a narrow-pipe hash function [12]. HAIFA modifies Merkle–Damgård by introducing extra input parameters to the compression function: a bit counter, and an optional salt value. The bit counter keeps track of the number of bits hashed so far. And the salt value is used as a key to create families of hash functions. Salt is set to 0 if only one hash function is required.

C. Sponge Construction

Sponge construction [13] is a new construction method for hash functions and stream ciphers. This construction can be built upon a function f which can be expressed as a random permutation or random function. If f is expressed as a random permutation, construction is called a P − sponge, otherwise, if it is expressed as a random function, construction is called a T − sponge. The main difference between the compression functions of MD structures and the f function in sponge construction is, unlike the compressing functions in MD or Haifa, f is a function which maps l bit input to l bit output. The construction is consisting of 2 phases: absorbing and squeezing. In the first phase, data is input to the sponge iteratively block by block and in the second phase output is given in the same manner. These two phases have similar procedures. In the absorbing phase, iteratively, message block is XOR–ed or overwritten to the state and f is applied to this state. Then, next message block is processed and so on. After all the message blocks are input, second phase is applied. In the squeezing phase, some part of the state is output and f is applied to the state. Then, again some part of the state is output and f is applied until the desired hash size is achieved. This process is depicted in Fig. 2. Optionally, between two phases, some number of blank rounds can be applied. In the blank rounds, there is no input to or output from the state. Only, f is applied to the state.

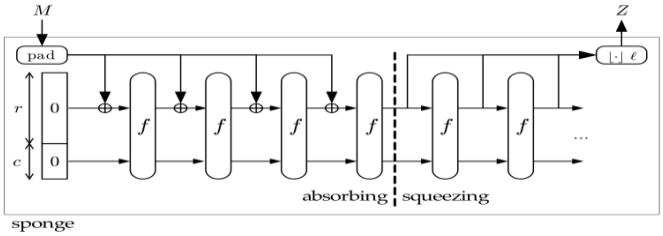


Fig. 2: Sponge Construction

The size of the inner part of the state of sponge construction is called the capacity of the sponge function and denoted as c. The security of a sponge construction depends on its capacity c, hash size n and f function. For P-sponges, the complexity of a collision is $\min(2^{c/2}, 2^{n/2})$, and complexity of pre image and second pre image is $\min(2^{c/2}, 2^{n/2})$. Collision complexity of T-sponges is equal to the collision complexity of P-collisions. Finding a pre image costs $\min(2^{c}, 2^{n})$,

and finding a second pre image costs min(2^c/L, 2^n) for a T-sponge, where L is the length of the original message. This construction ensures that if f is a random function and $c \geq 2n$, then the sponge construction is in differentiable from a random oracle.

IV. SHA FAMILY

A secure hash algorithm is considered to be cryptographically secure. The original data, once hashed by a secure hash algorithm, typically cannot be reconstructed with a feasible amount of computing power. Some network routers and firewalls implement secure hash algorithms directly in their hardware. This allows data packets to be authenticated with limited impact on throughput. Secure hash algorithm software exists also, including many open source implementations. The US National Institute of Standards and Technology (NIST) and the Canadian Communications Security Establishment (CSE) jointly run the Cryptographic Module Verification Program (CMVP).

The US government has standardized at least six secure hash algorithms. SHA-0 and SHA-1 were the earliest incarnations developed in the 1990s. The SHA-2 series developed in the 2000s included SHA-224, -256, -384 and -512. These are designed such that two documents with different contents generally produce two unique sets of hash values, avoiding hash collisions.

The SHA-0 algorithm, first published in 1993 by the NIST, was quickly discontinued after a significant weakness was found. It was replaced by SHA-1 in 1995, which includes an extra computational step that addresses the undisclosed problems of SHA-0. Both algorithms hash a message of up to 2^{64}-1 bits into a 160-bit "digest." Both utilize a block size of 512 bits and a word size of 32 bits in their operation.

SHA-1 is used in some common Internet protocols and security tools. These include IPsec, PGP, SSL, S/MIME, SSH and TLS. SHA-1 is also typically used as part of the protection scheme for unclassified government documents. The private sector utilizes this secure hash algorithm for some sensitive information as well. It is scheduled to be retired from general government use in 2010.

SHA-224, -256, -384 and -512 were published by the NIST between 2001 and 2004. These four algorithms, also known as the SHA-2 family, are generally more robust than SHA-1. SHA-224 and SHA-256 utilize the same block, word and maximum input message sizes as SHA-1. In contrast, SHA-224 produces a 224-bit digest, while SHA-256 creates a 256-bit digest. SHA-384 and SHA-512 increase the block size to 1024 bits, the word size to 64 bits, and the maximum input message length to 2^{128}-1 bits. The digest produced by SHA-384 is 384 bits long, while the SHA-512 digest contains 512 bits. Like SHA-0 and SHA-1, the SHA-2 family was designed by the US National Security Agency (NSA). Although serious flaws have not been publicly disclosed in SHA-2, NIST has opened a competition to develop the next secure hash algorithm. This new algorithm, to be called SHA-3, is chosen in 2012 from a collection of public entries and announced keccak as the winner. It supports the same hash lengths as SHA-2, and its internal structure differs significantly from the rest of the SHA family. Table 1 gives the comparison of hash functions in SHAfamily.

Table 1: Comparison of SHA functions

Algorithm and Variant		Output Size (bits)	Internal State Size (bits)	Block Size (bits)	Max Message Size (bits)	Word Size (bit)	Rounds	Operations	Collisions found	Example Performance (MiB/s)
MD5		128	128	512	$2^{64}-1$	32	64	+,and,or,xor,rot	Yes	255
SHA-0		160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rot	Yes	-
SHA-1		160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rot	Theoretical attack (2^{61})	153
SHA-2	*SHA-224* *SHA-256*	224 256	256	512	$2^{64}-1$	32	64	+,and,or,xor,shr,rot	None	111
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512	1024	$2^{128}-1$	64	80	+,and,or,xor,shr,rot	None	99
SHA-3	224/256/384/512	1600			64	120		None		

V. CONCLUSIONS

An effort has been made to understand the various cryptographic hash functions. The focus of discussion was on study of different hash functions in SHA family. NIST announced a public competition in 2007 to develop a new cryptographic hash algorithm, to choose "SHA-3". Five finalists of this SHA-3 competition are *BLAKE, Grøstl, JH, Keccak,* and *Skein*. and announced keccak as the winner. It supports the same hash lengths as SHA-2, and its internal structure differs significantly from the rest of the SHA family. Keccak has higher performance in hardware implementations than SHA-2 or any of the other finalists [14]. An attack that could work on SHA-2 most likely would not work on Keccak because the two algorithms are designed so differently. SHA-2 has held up well and NIST considers SHA-2 to be secure and suitable for general use. It may take years to identify all the possibilities for Keccak, it immediately provides an essential insurance policy in case SHA-2 is ever broken.

REFERENCES

1. Krystian Matusiewicz, Analysis of Modern Dedicated Cryptographic Hash Functions, PhD thesis. Macquarie University, 2007.
2. Xiaoyun Wang and Hongbo Yu, How to Break MD5 and Other Hash Functions, EUROCRYPT, pp. 19–35, 2005.
3. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, Cryptanalysis of the Hash Functions MD4 and RIPEMD, EUROCRYPT, pp. 1–18, 2005.
4. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, Finding Collisions in the Full SHA–1, CRYPTO, pp. 17–36, 2005.
5. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin, Efficient Collision search Attacks on SHA–0, CRYPTO, pp. 1–16, 2005.
6. Menezes, Oorschot, and Vanstone, Handbook of Applied Cryptography, First Edition, CRC Press, Florida, 1997.
7. Praveen Gauravaram and Lars R. Knudsen, Cryptographic Hash Functions,. Handbook of Information and Communication Security, First Edition, Springer, pp. 59-79. 2010
8. Ralph C. Merkle, One Way Hash Functions and DES, CRYPTO, pp. 428–446, 1989.
9. Ivan Damgard, A Design Principle for Hash Functions, CRYPTO, pp. 416–427, 1989.
10. Eli Biham and Orr Dunkelman, A Framework for Iterative Hash Functions: HAIFA, Proceedings of Second NIST Cryptographic Hash Workshop.
11. John Kelsey and Bruce Schneier, Second Preimages on n-Bit Hash Functions for Much Less than 2n Work, EUROCRYPT, Springer, pp. 474–490, 2005.
12. Orr Dunkelman and Eli Biham, A Framework for Iterative Hash Functions: HAIFA, Second NIST Cryptographic Hash Workshop, Santa Barbara, Aug 24–25, 2006.
13. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche, Sponge Functions, 2007.
14. http://www.nist.gov/itl/csd/sha-100212.cfm.
15. http://en.wikipedia.org/wiki/SHA-1/SHA-2

AUTHORS PROFILE

B. Madhuravani, Assistant Professor, Department of CSE, MLR Institute of Technology, Dundigal,Hyderabad. She is doing Ph.D in ComputerScience & Engineering, JNTUH. Her research interests include Computer Networks, Network Security, Distributed Systems and Data Structures.

Dr. D. S. R. Murthy obtained **B. E. (Electronics)** from Bangalore University in 1982, **M. Tech. (CSE)** from Osmania University in 1985 and **Ph.D. (CSE)** from JNTUH, Hyderabad in 2011. He is presently working as **Professor in Information Technology**, SNIST, Hyderabad since Oct. 2004. He earlier worked as Lecturer in CSE, NIT (formerly REC), Warangal, India during Sep. 1985 – Feb. 1993, as Assistant Professor in CSE, JNTUCE, Anantapur, India during Feb. 1993 – May 1998, as Academic Coordinator, ISM, Icfaian Foundation, Hyderabad, India during May 1998 – May 2001 and as Associate Professor in CSE, SNIST during May 2001 - Sept. 2004. He worked as Head of the Dept. of CSE, JNTUCE, Anantapur during Jan. 1996 – Jan 1998, Dept. of IT, SNIST during Apr. 2005 – May 2006, and Oct. 2007 – Feb. 2009. He is a **Fellow** of **IE (I)**, **Fellow** of **IETE**, **Senior Life Member** of **Computer Society of India**, Life Member of ISTE, Life Member of SSI, DOEACC Expert member, and Chartered Engineer (IE (I) & IETE). He is Chief Superintendent of Examinations (Autonomous), SNIST, Hyderabad since February 2011. He is **Board of Studies Member** of SRKPG College (Autonomous), Nandyal, SKD University since August 2010 and Anurag Group of Institutions (Autonomous), Hyderabad, JNTUH since July 2012. He is a **Reviewer** of International Journal of Advanced Research in Computer Science (**IJARCS**), International Journal of Computational Intelligence and Information Security (**IJCIIS**), International Journal of Computer Science and Information Technology (**IJCSIT**), International Journal of Advanced Computer Science and Applications (**IJACSA**), Science Journal of Electrical & Electronic Engineering (**SJEEE**), International Journal of Network Security (**IJNS**), International Journal of Computer and Information Technology (**IJCIT**), **Honorary Peer Reviewer** of Global Journal of Researches in Engineering (**GJRE), Editorial Board Member** of World Academy of Research in Science and Engineering (**WARSE**), International Journal of Emerging Trends & Technology in Computer Science **(IJETTCS),** Journal of Computer Science and Engineering (**JCSE**), International Journal of Computer Applications and Information Technology (**IJCAIT**). He is a **Member** of International Association of Computer Science and Information Technology (**IACSIT**). He published a text book on C Programming & Data Structures. His **research interests** are **Image Processing, Image Cryptography and Network Security**. He published **research papers** in International Journal of Computer and Network Security (**IJCNS**), International Journal of Computer Theory and Engineering (**IJCTE**), International Journal of Computational Intelligence and Information Security (**IJCIIS**) and in International Journal of Advanced Research in Computer Science (**IJARCS**).