IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 37-42
www.iosrjournals.org

An overview of Cryptographic Hash Functions MD-5 and SHA

H. B. Pethe[1], Dr. S. R. Pande [2]
[1]*(Department of Electronics & Comp. Sc, RTMNU, Nagpur (India))*
[2]*(Department of Computer Science, SSESA's Science College, Nagpur.(India))*

Abstract: *Data Integrity is the most important service of cryptography. Hash functions are used to check the integrity of the file. Hash functions are useful, if the active attack is performed by the attacker. For the secured communication before sending any file though the unsecured network the hash value of any file or message is found out using hash function. After receiving the file by the recipient again the hash value of the received file is calculated. The hash value before sending the file and after sending the file is checked, if the hash value is same, then we can come to know that, the file is not changed. There are various algorithms for finding the hash value of any file. This paper gives an overview of hashing functions such as MD-5 and SHA that are used to maintain the integrity of file.*
Keywords: *Data Integrity, Cryptography, Hash functions, SHA, MD-5, File integrity.*

I. Introduction

The functions that produce a digital fingerprint of messages or data are found out by using hash functions. These functions are ubiquitous in today's IT systems and have a wide range of applications in security protocols and schemes, such as providing software integrity, digital signatures and password protection. Furthermore, hash function algorithms are also used for constructing pseudo-random number generators, key derivation algorithms, message authentication codes, as well as stream ciphers and block ciphers [1].

1.1 Hashing

Using Hashing a string of characters with variable length turns into a fixed-length value which represents the original string. In cryptography, hashing is a one-way operation which transforms a stream of data into a more compressed form called a message digest. The operation is not be invertible, meaning that recovering the original data stream from the message digest should not be possible. All the message digests or hash values generated by a given hash function have the same size no matter what the size of the input value is. The size of the hash value is depends on the algorithm used. On the other hand, encryption is a two-way operation which transforms a plaintext into a ciphertext and allows for the process to be inverted by transforming the ciphertext back into its original plaintext via a mechanism called decryption. Both operations depend on a key.

Cryptology is divided into two branches: cryptography and cryptanalysis. Cryptanalysis is the reverse process of cryptograph. Cryptography is the science that aims at designing and developing cryptographic systems, sometimes referred to as a cryptosystems. A cryptosystem is a set of methods needed to create a particular encryption and decryption scheme. A typical cryptosystem is made up of three parts: One that generates the encryption/ decryption key, one that performs the encryption process, and one that deals with the decryption process. Encryption is the process by which one changes a message (called plaintext) into unreadable text called ciphertext. Decryption is the inverse process which recovers the plaintext from the ciphertext

II. Cryptographic Hash Functions

The term hash function has been used in computer science from quite some time and it refers to a function that compresses a string of arbitrary input to a string of fixed length. However if it satisfies some additional requirements, then it can be used for cryptographic applications and then known as Cryptographic Hash functions. Cryptographic Hash functions are one of the most important tool in the field of cryptography and are used to achieve a number of security goals like authenticity, digital signatures, pseudo number generation, digital steganography, digital time stamping etc.[2]

A hash function, is a function that takes some message of any length as input and transforms it into a fixed-length output called a hash value, a message digest, a checksum, or a digital fingerprint. A hash function is a function
f : D -> R, where the domain D = $\{0,1\}^*$, which means that the elements of the domain consist of binary string of variable length; and the range R = $\{0,1\}^n$ for some n >= 1, which means that the elements of the range are

binary string of fixed-length. So, f is a function which takes as input a message M of any size and produces a fixed-length hash result h of size n. A hash function f is referred to as compression function when its domain D is finite, i.e. if the function f takes as input a fixed-length message and produces a shorter fixed-length output[3].

Security properties of cryptographic hash function (H) :

1. H should accept a block of data of any size as input.
2. H should produce a fixed-length output.
3. H should behave like random function while being deterministic and efficiently reproducible. H should accept an input of any length, and outputs a random string of fixed length. For the same input, H should always produce the same output.
4. If the message M is given, it is easy to compute its corresponding digest h; where h can be computed in polynomial time O(n) where n is the length of the input message.
5. Given a message digest h, it is computationally difficult to find M such that H(M) = h. This is called the one-way or pre-image resistance property i.e. It is not possible to recover message from the given hash value.
6. Given a message M1, it is computationally infeasible to find another message $M_2 \neq M_1$ with $H(M_1) = H(M_2)$. This is called the weak collision resistance or second pre-image resistance property [4].
7. It is computationally infeasible to find any pair of distinct messages (M1, M2) such that $H(M_1) = H(M_2)$. This is referred to as the strong collision resistance property [5].

III. Cryptographic Hash Algorithms

Cryptographic hash functions come in different shape and size. There are basically two main categories of hash functions. Hash functions that depends on a key for their computation, usually known as Message Authentication Code or MAC and hash functions that do not depend on a key for their computation, generally known as un-keyed hash function or simply hash function. All well known hash functions are either based on a block cipher or on modular arithmetic.

3.1 MD 5 Hash Algorithm

The MD5 (1992) message-digest algorithm was designed as a strengthened extension of the MD4 (1990) message digest algorithm. MD5 is slightly slower than MD4. Both algorithms were developed by Ron Rivest.

The algorithm accepts an input message of arbitrary length and produces a 128-bit "Message digest", fingerprint" or "hash result". The following fig. shows the way the input message is turned into a 128-bit message digest [6].

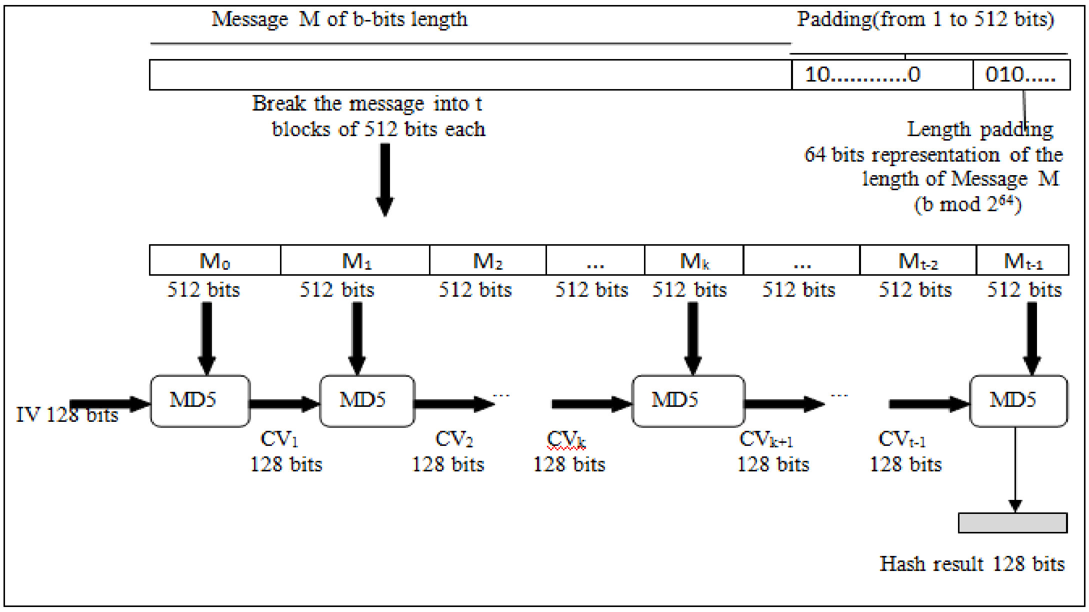

Figure: The MD-5 Algorithm

The actual processing of the MD5 algorithm consists of the following 5 steps:

Step 1: Append padding bits

The message is extended or padded in such a way that its total length in bits is congruent to 448 modulo 512. This operation is always performed even if the message's length in bit is originally congruent to 448 modulo 512. 448 + 64 = 512, so the message is padded such that its length is now 64 bits less an integer multiple of 512. Padding is done by appending to the message a single “1" bit followed by the necessary amount of “0" bits so that the length in bits of the padded message becomes congruent to 448 modulo 512. For example, if the message is 447 bits long, it is padded by 1 bit to a length of 448 bits (the single bit 1 is appended to the end of the message in this particular case). On the other hand, if the message is 448 bits long, it is padded by 512 bits to a length of 960 bits. Thus, at least 1 bit and at most 512 bits are appended during this step.

Step 2 : Append length

A 64 bit representation of the length in bits of the original message M (before the padding bits were added) is appended to the result of step 1. If the length of the original message is greater than 2^{64} (= 184 467 440 73 709 551 616), then only the low order 64 bits of the length of message M are used. Hence, the field contains the length of the original message M modulo 2^{64}. These bits are appended as two 32 bit words and appended low-order (least significant) word first. The result of step 1 and step 2 is a message with a length in bits that is an integer multiple of 512 bits.
Suppose that we are given a message M with a length in bit of 447.
Step 1 tells us that a single bit will be added to that message to bring its length to 448.
Step 2 appends a 64-bit representation of the length of the original message (447) which is less than 264 and we append
the following 64-bit representation of the decimal number 447:
0000000000000000000000000000000000000000000000000000000110111111

Step 3: Initialize MD buffer

A 128 - bit (4 X 32 - bit) buffer (A, B, C, D) is used to hold intermediate and final result of the MD5 hash algorithm. These registers are initialized to the following 32-bit values in hexadecimal:

A = 67 45 23 01
B = ef cd ab 89
C = 98 ba dc fe
D = 10 32 54 76

These values are stored in **little-endian format**, meaning that the low -order bytes of a word is placed in the low-address byte position. The initialization values appear then as follows:

word A = 01 23 45 67
word B = 89 ab cd ef
word C = fe dc ba 98
word D = 76 54 32 10

These four variables are copied into different variables: A is saved as AA, B as BB, C as CC and D is saved as DD.

Step 4: Define four auxiliary functions and process message

This step consists of sixty-four (64) steps divided into four (4) rounds of processing. The four rounds are almost identical, with the main difference being that each round uses a different primitive logical function, denoted by F, G, H, and I in Table 1. We note that each of them takes three 32-bit words as input and yields one 32-bit word as output.

Table 1: The Primitive Functions Of The MD5 Compression Algorithm.

Round	Primitive function	Steps (64)
1	$F(X,Y,Z) = (X \wedge Y) \vee (\sim X \wedge Z)$	$0 \le j \le 15$
2	$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \sim Z)$	$16 \le j \le 31$
3	$H(X,Y,Z) = (X \oplus Y \oplus Z)$	$32 \le j \le 47$
4	$I(X,Y,Z) = (Y \oplus (X \vee \sim Z)$	$48 \le j \le 63$

Where,

$\wedge$ = AND, $\vee$ = OR, $\oplus$ = XOR, ~X = NOT(X)

Each round takes as input the current 512-bit message block M_k and the 128-bit buffer value ABCD and produces as output an updated value of the buffer earlier referred to as the chaining variable CV_k. In addition, each round also uses one-fourth of a 64-element table denoted T[1 ... 64] which is constructed from the sine function. It is constructed such that the i^{th} element of the table T, denoted T[i], is equal to the integer part of 232 x abs(sin(i)), where i is in radian. Since abs x (sin(i)) is a number between 0 and 1, the elements of the table T are numbers less than or equal to 2^{32}, Hence, they can be represented in 32 bits.

Step 5: Output

The output from the very last round is the 128-bit hash result or message digest we obtain after we have incrementally processed all t 512-bit blocks of the message. The entire process can be summarized as follows:
CV0 = IV
CV_{k+1} = SUM32(CV_k, RF_I [M_k, RF_H[M_k;RF_G[M_k,RF_F [Mk,CV_k]]]])
MD5SUM = CV_t
Where,
IV = the initial value of the ABCD buffer, defined by step 3
M_k = the k th 512-bit block of the message
CV_k = the chaining variable processed with the k th block of message
RFx = the round function using primitive logical function x
MD5SUM = the final hash result or message digest
SUM_{32} = addition modulo 2^{32}

3.2 The Secure Hash Algorithm - SHA

The Secure Hash Algorithm (SHA) was developed by the National Security Agency (NSA) and published in 1993 by the National Institute of Standard and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS PUB 180). SHA is based on and shares the same building blocks as the MD4 algorithm. The design of SHA introduced a new procedure which expands the 16-word message block input to the compression function to an 80-word block among other things. In 1994, NIST announced that a technical flaw in SHA was found. And, this flaw makes the algorithm less secure than originally believed. No further details were given to the public, only that a small modification was made to the algorithm which was now known as SHA-1 and published in FIBS PUB 180-1.

The SHA-2 family of hash algorithm consists of five cryptographic hash functions denoted by SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512[7-9]. The last four variants are sometimes collectively referred to as SHA-2. In fact, NIST updated its hash function standard to FIPS PUB 180-2 in 2002. This update specified three new hash functions, next to SHA-1, known as SHA-256, SHA-384, and SHA-512. SHA-1 is designed to produce a 160-bit message digest. The other hash functions have the length of their message digest indicated by the number following prefix “SHA-"; thus SHA-256 produces a 256-bit message digest, whereas SHA-384 produces a 384-bit hash value and so on.
SHA-224 which produces a message digest of 224-bit was added to the standard in 2004[5].
Description of the SHA-1 algorithm

The SHA-1 algorithm accepts as input a message with a maximum length of 2^{64}-1 and produces a 160-bit message digest as output. The message is processed by the compression function in 512-bit block. Each block is divided further into sixteen 32-bit words denoted by M_t for t = 0, 1, ... , 15. The compression function consists of four rounds; each round is made up of a sequence of twenty steps. A complete SHA-1 round consists of eighty steps where a block length of 512 bits is used together with a 160-bit chaining variable to finally produce a 160-bit hash value. The processing of SHA-1 works as follows :

Step 1: Append padding bits

The original message is padded so that its length is congruent to 448 modulo 512. Padding is always added although the message already has the desired length. Padding consists of a single 1 followed by the necessary number of 0 bits.

Step 2: Append length

A 64-bit block treated as an unsigned 64-bit integer (most significant byte first), and representing the length of the original message (before padding in step 1), is appended to the message. The entire message's length is now a multiple of 512.

Step 3: Initialize the buffer

The buffer consists of five (5) registers of 32 bits each denoted by A, B, C, D, and E. This 160-bit buffer is used to hold temporary and final results of the compression function. These five registers are initialized to the following 32-bit integers (in hexadecimal notation).

A = 67 45 23 01

B = ef cd ab 89
C = 98 ba dc fe
D = 10 32 54 76
E = c3 d2 e1 f0

The registers A, B, C, and D are exactly the same as the four registers used in MD5 algorithm. But in SHA-1, these values are stored in **big-endian format**, which means that the most significant byte of the word is placed in the low-address byte position. Hence the initialization values (in hexadecimal notation) appear as follows:

word A = 67 45 23 01
word B = ef cd ab 89
word C = 98 ba dc fe
word D = 10 32 54 76
word E = c3 d2 e1 f0

Step 4: Process message in 512-bit blocks

The compression function is divided into twenty sequential steps composed of four rounds of processing where each round is made up of twenty steps. The four rounds are structurally similar to one another with the only difference that each round uses a different Boolean function, which we refer to as f1, f2, f3, f4 and one of four different additive constants K_t ($0 \leq t \leq 79$) which depends on the step under consideration. The values of the four dictinct additives constant are given in table 3 below.

Step 5: Output

After processing the last 512-bit message block t (assuming that the message is divided into t 512-bit blocks), we obtain a 160-bit message digest.

IV. Applications of Cryptographic Hash Functions

Hash functions are used in many situations to support various cryptographic protocols. Their most common application is the creation and verification of digital signature.

4.1 Digital Signature

Hand-written signature is a way to prove that a paper document is signed by us and not by someone else. To prove this, the current hand-written signature is compared with one or more of our previous hand-written signatures. If there is a match then the recipient of the document can safely accept that the document could not have been signed by someone else. In case it is the first time, we have to prove our identity by means of some identification card, and necessarily by being physically present to sign the document.
Some properties of the hand-written signatures are:

- The signature should be unique to each person.
- The signature should be verifiable as belonging a particular person.

The digital signature is the electronic equivalent to the hand-written signature with regard to its purpose. More precisely, a digital signature is a sort electronic “stamp" or “digital fingerprint" placed on a document that is unique to the signer of the document and to the signed document. One major difference between a digital and a hand-written signature is that for every different document, the digital signature is different even if the signer and the private/public key pair are the same. A digital signature scheme provides both data integrity protection and data origin authentication.

4.2 File Integrity Verification

Hash functions are widely used to verify file integrity. Once downloaded free software on the Internet, have probably visited websites which publish the checksum of the software near the hyperlink of the binary executable file or the archived source code of the corresponding software. Without this vital piece of information which is the checksum of the software, one will have a hard time verifying the integrity of downloaded software.

4.3 Password Hashing

A password, in computer science, is a secret sequence of character that one uses to gain access to a file, an application or a computer system. Password has been used long before our time. It used to be a secret word or phrase which enabled a person to be accepted as a friend by soldiers posted to keep watch and guard. Password hashing was used since the early ages of the UNIX operating system. Users of UNIX systems have their password hashed and stored in a password file. Fortunately, some web applications generate a hash value of all passwords and store these hash values, rather than the password itself, in the database.

4.4 Key Derivation

Key derivation is the process of deriving various keys from a shared secret password or passphrase (which typically does not have the desired properties to be used directly as cryptographic keys) to secure a communication session. For example, two people can agree on a secret key and pass that key to a key derivation function to produce keys from encryption and authentication. This guarantees that an attacker who learns your authentication key will not have access to your encryption key.

4.5 Trusted Digital Time Stamping

It is desirable to bind a time together with a document as to certify its existence at that particular time. In the matter of intellectual property where dispute may arise between two or more people about who was the first one to make a discovery or an invention, time-stamping can play an important role at determining who is right.

4.6 Rootkit Detection

A Root kit is a program or a set of programs that a hacker installs on the victim's computer in order to cover the tracks of other malicious programs which attempt to corrupt an operating system. A Rootkit will hide its presence on a compromised system. Root kits can be detected in several ways including signature-based detection which uses scanning tools much like antivirus or antispyware programs that scan the system for signs of previously known rootkits patterns.

V. Conclusion

There are various applications of Hash functions, but the main application is to check the integrity of file. The interesting property that, every bit of the output is a function of every bit of the input. MD5 is strong for a 128 bit hash code. Finding two messages having the same hash value requires 2^{64} operations. SHA-1 is performed modulo $2^{32,}$ thus it is well suited for a 32 bit architecture. Because SHA-1 involves 80 steps and must process 160 bit buffer, it is slower than MD5 on the same architecture. SHA-1 is also simple to describe and to implement in both software and hardware.

References

[1]. Liu Jian – dong, Tian Ye, Wang Shu-hong, Yang Kai, "A fast New one way cryptographic hash function", *IEEE 2010.*

[2]. Joseph Sterling Grah "Hash functions in cryptography"

[3]. G. Bertoni, J. Daemen, M. Peeters, & G. V. Assche (2012), Keccak An update. Retrieved March 22-23, 2012, from Third SHA-3 candidate conference, Washington DC.

[4]. Danilo Gligoroski, Svein Johan Knapskog, J0rn Amundsen, Rune Erlend Jensen "Internationally Standardized Efficient Cryptographic Hash Function".

[5]. Abdulaziz Ali Alkandari, Imad Fakhri Al-shaikhli, Mohammad A. Alahmad "Cryptographic Hash functions: A High Level View" *International conference on informatics and creative multimedia*.

[6]. I. Damgård. A design principle for hash functions. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings,* volume 435 of *Lecture Notes in Computer Science,* pages 416– 427. Springer, 1990.

[7]. FIPS 180, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, Washington D. C., 1993.

[8]. FIPS 180-1, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, Washington D. C.,1995.

[9]. FIPS 180-2, Secure Hash Standard (SHS), National Institute of Standards and Technology, US Department of Commerce, Washington D. C.,2002.