A Review on Hashing Function- Message Digest

Renuka S. Durge[1]
Computer Science and Engineering,
Prof. Ram Meghe Institute of Technology and Research, Badnera
Amravati.,Maharashtra.
renuka434@gmail.com[1]

Dr. Vaishali M. Deshmukh [2]
Computer Science and Engineering,
Prof. Ram Meghe Institute of Technology and Research, Badnera
Amravati.,Maharashtra.
vmdeshmukh@mitra.ac.in[2]

Abstract.-- The development of information encryption technology is important and vital in securing data to give the network's rising popularity. Information must be protected in order to maintain its validity and integrity and enable secure communication between communicative parties. The hash function is at the heart of contemporary cryptography, and the MD5 series of algorithms are the most widely used hash algorithms. Cryptographic hash functions are crucial to the study of cryptography One of the most crucial tools in the science of cryptography, cryptographic hash functions are used to accomplish a variety of security objectives, including authenticity, digital signatures, pseudo number generation, digital steganography, and digital time stamping. This paper emphasises the importance of hash functions its varied structures, design approaches attacks, and the consistent recent growth in this field.

Keywords*: *Cryptographic Algorithms, Hash function , Message Digest, SHA, Digital signature.

I. INTRODUCTION

Information concealment is essentially the goal of cryptography. Cryptography is used to secure computer passwords and move data between systems. It is the science of encrypting and decrypting data using mathematical formulas. It allows for the storage or transmission of sensitive data via unsecure networks so that only the intended recipient can access it. Thus it is the art and science of hiding information during transmission by transforming it into an unrecognizably different form [1]. The technique of scrambling data material, including text, photos, audio, and video, to make it inaccessible, invisible, or unintelligible during transmission or storage is known as data cryptography, sometimes known as encryption. Data protection from unauthorized attackers is the main goal of cryptography. The reverse of data encryption is data decryption.Plaintext refers to the original material that will be transferred or stored; this data may be read and understood by both a person and a computer [2]. Contrarily, the data is referred to as cypher text because it cannot be read by either humans or machines. Cryptosystem refers to a system that offers encryption and decryption. A cryptosystem uses an encryption algorithm to encrypt and decrypt data. The algorithm dictates how simple or complex the encryption process will be, the required software, and the key. The size of an encryption algorithm's key space is a gauge of how secure it is [3]. Key space refers to the range of possible values for keys.

The harder an attacker will have to work to crack the key, the higher the security level of such an algorithm will be. The key in encryption is a piece of data that determines how plaintext will be converted to cypher text or reversed during decryption. More possible keys can be created if the key space is greater. The key size of 256, for instance, would result in a 2256 key space if the key sizes were 128, 192, or 256 bits. The secrecy of the key, key length, initialization vector, and how they interact determine the encryption algorithm's strength.

II. METHODOLOGY

In this work main focus will be on deep review of Message digest algorithm, which will be helpful for better understanding in working of cryptographic algorithm.

A. MESSAGE DIGEST

A hash function used in cryptography is called Message Digest 2. Ronald Rivest developed it in 1989, and it generates a 128-bit hash value from a message of any length. It was created with 8-bit computing systems in mind. The main purpose of Message Digest 2's development was to be used in digital signature applications, which required signing a sizable, encrypted file with a private key. It is still used in public key infrastructures, but because it is computationally expensive and is no longer regarded as secure, it is rarely employed.

The Message Digest 2 method allows for the use of messages of any length and produces a 128-bit message digest of the input as its output. According to the presumption, it is impossible for two messages to have the same message digest or for a message to have a target message digest of a special kind.

The Message Digest 2 algorithm includes the following steps: appending padding bytes, appending checksum, initialising the message digest buffer for computing the message digest, processing the message in blocks of 16 bytes, and producing the output at the conclusion of the process.

One of Message Digest 2's key benefits is how simple it is to use. Message Digest 2 is slower than Message Digest 4 or 5. The ease of use of Message Digest 2 is one of its main advantages. In contrast to Message Digest 4 or 5, Message Digest 2 is slower.

This is because it was created for 8-bit computers, whereas Message Digest 4 and 5 were optimised for 32-bit systems. Again, compared to secure hash algorithms like SHA-1 or SHA-256, Message Digest 2 techniques operate more slowly. Despite this, Message Digest 2 is no longer widely used because it was discovered that crucial information could be revealed through collision attacks. The static function getInstance is used to initialise these algorithms (). After selecting an algorithm, the message digest value is calculated, and the results are sent in a byte array. The BigInteger class is used to convert the output byte array into its signum representation. This representation is then transformed into a hexadecimal format to yield the necessary Message Digest.
Example:

```
importjava.math.BigInteger;
importjava.security.MessageDigest;
importjava.security.NoSuchAlgorithmException;
 publicclassGFG {
  publicstaticStringencryptThisString(String input)
  {
     try{
        // getInstance() method is called with algorithm
MD2
        MessageDigest md =
MessageDigest.getInstance("MD2");
        byte[] messageDigest =
md.digest(input.getBytes());
        BigInteger no = newBigInteger(1, messageDigest);
        String hashtext = no.toString(16);
        while(hashtext.length() <32) {
           hashtext = "0"+ hashtext;
        }
        returnhashtext;
     }
     catch(NoSuchAlgorithmException {
        thrownewRuntimeException(e);
     }
  }
  publicstaticvoidmain(String args[]) throws
                              NoSuchAlgorithmException
  {
     System.out.println("HashCode Generated by MD2 for:
");
        String s1 = "hello world";
     System.out.println("\n"+ s1 + " : "+
encryptThisString(s1));
  }
}
HashCode Generated by MD2 for:
hello world : d9cce882ee690a5c1ce70beff3a78c77
```

B. MD4

Ronald Rivest, an MIT professor, developed MD4, the fourth in a line of message digest algorithms. It is implemented to employ a cryptographic hash function for message integrity checks. There are 128 bits in the digest. The technique had an impact on later designs including the MD5, SHA, and RIPEMD algorithms. Microsoft Windows NT, XP, and Vista also use MD4 to compute NT-hash password digests.Because MD4 was intended to be quick, some security risks had to be taken. Since 1992, vulnerabilities had been discovered, which prompted Rivest to create MD5, a more robust but slower version. Dobbertin [1, 2] discovered the first MD4 collisions in year 1998 that gives a strategy that is producing them, with this work it is equivalent to computing factor in 220 MD4 hashes.

The MD4 algorithm is composed of five steps:

a) Append using Padding bits

This step is quite simple; we simply lengthen the input message until its bit length is equal to 448 modulo 512. As stated in the RFC, we must produce a message whose most recent block is 64 bits short of 512 bits. The padding pattern is as follows: a bit of 1 immediately following the message, followed by a string of 0s to get to 448 modulo 512.

b) Append Length

At this point, the input message is being formatted instead of the hash process itself. To make our input message the ideal size of 512 bits by blocks, we will now fill the final 64 bits of the previous block. Next, we will add a representation of the message's length before padding to our processed input message.

c) *Initialize MD buffer*

The message digest will be constructed by incrementing four variables (each with a size of four bytes and a word size of 32 bits), which must first be created with the following precise values:
'Magic' values: word A = 0x67452301, word B = 0xefcdab89, word C = 0x98badcfe, and word D = 0x10325476
• The initialization of word A is 0x01, 0x23, 0x45, and 0x67 reversed.
Word B has the inverted values 0x89, 0xAB, 0xCD, and 0xEF (prolonging A).
• The inverted values 0xFE, 0xDC, 0xBA, and 0x98 are being applied to word C.
• The letters 0x76, 0x54, 0x32, and 0x10 in the word D are reversed.

Briefly put, the functions perform the following bitwise operations:
• F: if X, Y, or Z, then
• G: "1" if at least two of X, Y, or Z exist; else, "0."
• H: Z or X or Y or

The aforementioned boolean table will be applied to every word's input component.

d) *Execute the message in blocks of 16 words.*

Before moving on, let's define a few auxiliary functions that will help with the computations listed below.

The three functions that follow accept three words as input and output one word after performing certain binary operations. To keep the result under 32 bits, keep in mind that each word addition will be performed while maintaining the word's integrity. To achieve this, we shall apply % 232 to each addition. This will only require moduling the output with % 0x100000000 from a coding perspective. Any number greater than 11111111 11111111 11111111, or 4294967295 in decimal form, will be reset to zero.

We are now prepared to process each block of 512 bits. In order to continue with our computation, we will first parse each 512-bit string (64 characters) into a 16-word array. We shall convert each range of four characters (or four bytes) into an integer representation in order to achieve this.

e) *Output*

The next step is to decode each of the four 32-bit MD variables back into four bytes before presenting the digest in hexadecimal. Rivest provides a description of the MD4 method and an effective implementation in RFC 1320.

MD4 uses 32-bit words to operate. Let M represent the hashed message. The padding applied to the message M results in a message length (in bits) same as 448 modulo 512, or sixty four bits much less than a multiple of 512. A single 1 bit precedes a a enough quantity of zeros to pad the message to the essential length.Even if the length of M happens to be 448 mod 512, padding is always applied. As a result, there is padding of at least one bit and a maximum of 512 bits. The message's length (in bits, before padding) is then added as a 64-bit block. In addition to being a multiple of 512 bits, the padded message is also a multiple of 32 bits. Let N be the total amount of 32-bit words in the message, and let M represent the message. N is a multiple of 16, as a result of the padding.

C. MD5

The cryptographic hash function algorithm MD5 accepts messages of any length and transforms them into messages of a predetermined length of 16 bytes. The MD5 algorithm is another name for the message-digest technique. MD5 was developed as an improvement on MD4 with more complex security goals. The output of the digest size (MD5) is always 128 bits. In 1991, Ronald Rivest developed MD5.

- **Use Of MD5 Algorithm:**

• File authentication uses it.
• It is utilised for security in web applications. For instance, secure user passwords, etc.
• We can store our password in a format of 128 bits using this approach.[6]

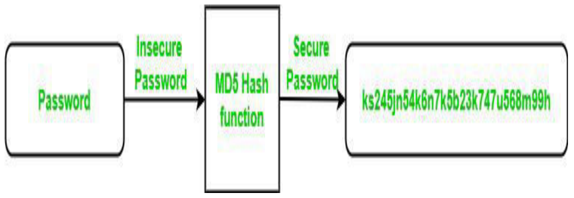

Fig 1. Basic Working Encryption Technique.

- **Working of the MD5 Algorithm:**

MD5 algorithm follows the following steps in figure 1.

a) *Append Padding Bits*:

In the first stage, we modify the original message by inserting padding bits so that the total message length is 64 bits less than the precise multiple of 512 [5].Consider receiving a message with 1000 bits. We now need to modify the original message by adding padding bits. Here, we shall padded the original message with 472 bits. The original message/output of the first step will be 1472 after the padding bits have been added, which is 64 bits less than an exact multiple of 512 (i.e., 512*3 = 1536).

Length(original message + padding bits) = 512 * i – 64 where i = 1,2,3 . . .

b*) Append Length Bits*:

In order to make the total number of bits in this step a perfect multiple of 512, we first add the length bit to the output of the previous step. We only add the 64-bit length bit in this case to the result of the first step.

i.e., the output of the first step is equal to 512 * n - 64 length bits.
We shall have 512 * n, which is an exact multiple of 512, after adding the two.

c) *Initialize the MD buffer*: Here, we employ the J, K, L, and M buffers. Each buffer has a 32-bit size.

- J = 0x67425301, K is equal to 0xEDFCBA45, L is 0x98CBADFE, and M is 0x13DCE476.

d) Process Each 512-bit Block: The MD5 algorithm's most crucial phase is this one. 64 procedures are performed in total throughout 4 cycles in this scenario. The first round will consist of 16 operations, the second round will consist of 16 operations, the third round will consist of 16 operations, and the fourth round will consist of 16 operations. The F function, for instance, is used for the first round, followed by the G function, the H function, the third round, and the I function, the fourth round.We employ logic gates like as OR, AND, XOR, and NOT to calculate functions. For each function, we use K, L, and M as our three buffers.
- K AND L OR F(K,L,M) (NOT K AND M)

- K AND L OR G(K,L,M) (L AND NOT M)

H(K,L,M) is equal to K XOR L XOR M.

- L XOR = I(K,L,M) (K OR NOT M)
Now that the function has been applied, we are going to do something with each block. We require the following to accomplish operations: add modulo 232
• 32-bit message M[i].
• The 32-bit constant K[i].
• Left shift by an amount of n bits.

Now, assume J, K, L, and M as input to initialise the MD buffer. K's output will be put into L, which will then feed into M, who will then feed J. After accomplishing this, we now run a few procedures to determine J's output.
• The function F is first applied to the outputs of K, L, and M in the first step. The output of this will be modulo 232 bits added with J.
• The M[i] bit message is added to the first step's output in the second step.
• After that, add the 32-bit constant K[i] to the second step's output.
• Finally, we do addition modulo by 232 and shift left by n (which may be random number of n).
The outcome of step J will be put into step K after all steps. The same procedures will now be followed for all G, H, and I functions. We shall obtain our message digest once all 64 operations have been completed.

e) Output:

The buffers J, K, L, and M, beginning with the lower bit J and ending with the higher bit M, hold the MD5 output once all rounds have been completed.

f) Process of MD 5

a) Steps :

1. The MD5 technique will be used to verify file authentication in Figure 2.
2. The input file is converted into 512 blocks, all data is compressed into 128 bits, the data is divided into 4 blocks of 32 bits, and each block is subjected to binary shifting. Hex decimalize each block; aggregate all blocks into a 128-bit hash value.
3. Each page is given a hash value via MD5. Two times are used to generate the hash value. sending a file for the first time. When a different user received the file a second time. If each hash value is different from the other, the file has been updated.

b) Applications of MD5 Algorithm:

- To check the authenticity of files and ensure their integrity, we use message digests.
- For data security and encryption, MD5 was utilised.
- It is employed for both password verification and message digestion, regardless of message size.
- For graphics and game boards.

c) Benefits of the MD5 Algorithm:

- MD5 gives quick as well as easy comprehend.
- A strong password is generated by the MD5 algorithm in a 16-byte format. To safeguard user passwords, all developers, including web developers, employ the MD5 algorithm.
- Very little RAM is required to integrate the MD5 algorithm.
- Generating a digest message from the original communication is simple and quick.

III. COMPARATIVE ANALYSIS

Due to the lack of complexity in MD4's hash calculation, it was deemed insecure. Although MD4 hashesresemble MD5 hashes, MD5 has much more going on in the background because the computation had many more steps added to it to make it more sophisticated. For many years, MD5 was safe enough to use, however nowadays it lacks the complexity needed for data encryption and cryptography. A new standard is required since powerful enough computers can now quickly decrypt MD5 hashes. A hashing algorithm needs to be in the "sweet spot" of complexity — neither too simple nor too complicated that it becomes difficult to use. With present computing capability, MD5 collisions are simply too trivial to obtain. The security applications of hash algorithms are revived when SHA (Secure Hash Algorithm) fixes the flaws in MD5. The two variants may easily be distinguished since SHA produces a longer string of hexadecimal characters.

This is a major factor in why SHA is more secure, as complexity rises exponentially as bit count grows. In this situations when security time is concern, MD5 is enough and still a fantastic choice for data verification.

IV. CONCLUSION

Thus hashing algorithm MD5 generates an input by hash value. Also offers securing messages and a reliable means of verifying the accuracy of any data transferred between servers. This suggested MD5 algorithm has been improved to increase collision resistance and provide access to the greatest levels of data security that makes it more difficult for hackers and attackers to forecast and alter specific data.

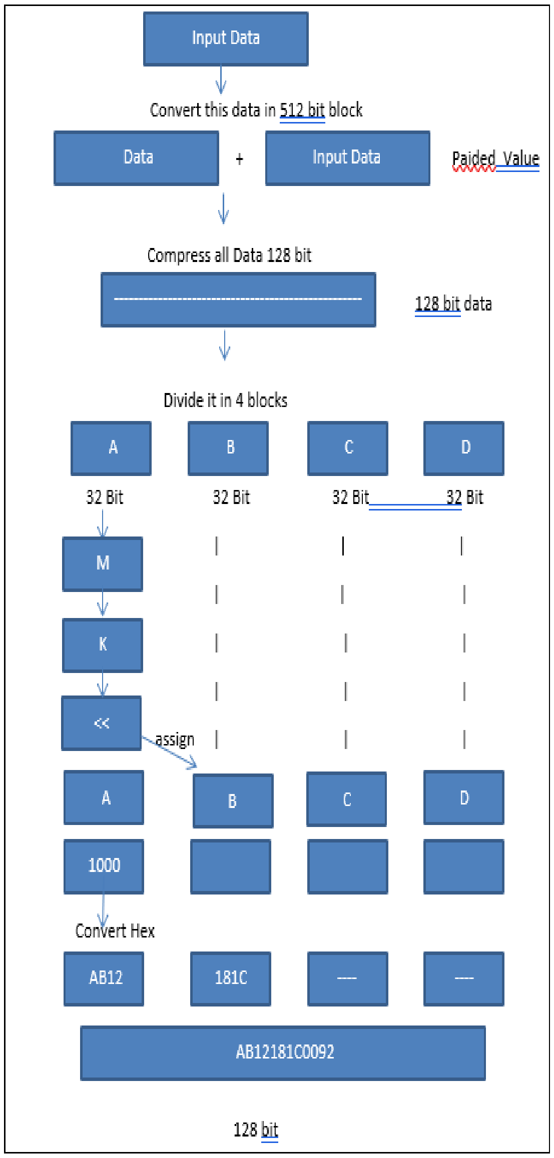

Fig 2. Working of Message Digest 5

References

1. H. Dobbertin, "*Cryptanalysis of MD4"*, Proceedings of the Third International Workshop on Fast Software Encryption, LNCS 1039, D. Goll- mann, Ed., Springer-Verlag, 1996, pp. 53-69
2. H. Dobbertin, "*Cryptanalysis of MD4"*, Journal of Cryptology, Vol. 11, NO. 4, 1998, pp. 253-271.
3. Wikipedia, MD5, Availableat: https:/en.wikipedia.org/wiki/MD5. (2015) Search Security, MD5v Definition
4. http://searchsecurity.techtarget.com/definition/MD 5. (2005).
5. Iusmentis, "The MD5 cryptographic hash function", Available at: http://www.iusmentis.com/technology/hashfunctio ns/md5/. (2005)
6. Srikanth Ramesh, "What is MD5 Hash and How to Use it", Available at:http://www.gohacking.com/ what-is-md5-hash/. (2009).