**Description of the MD5 Algorithm**

Takes as input a message of arbitrary length and produces as output a 128 bit “fingerprint” or “message digest” of the input.

It is conjectured that it is computationally infeasible to produce two messages having the samemessage digest.

Intended where a large file must be “compressed” in a secure manner before being encrypted with a private key under a public-key cryptosystem such as PGP.

Suppose a b-bit message as input, and that we need to find its message digest.

**Step 1 – append padded bits:**

– The message is padded so that its length is congruent to 448, modulo 512.

• Means extended to just 64 bits shy of being of 512 bits long.

– A single “1” bit is appended to the message, and then “0” bits are appended so that the length in bits equals 448 modulo 512.

**Step 2 – append length:**

– A 64 bit representation of b is appended to the result of the previous step.

– The resulting message has a length that is an exact multiple of 512 bits.

**Step 3 – Initialize MD Buffer**

• A four-word buffer (A,B,C,D) is used to compute the message digest.

– Here each of A,B,C,D, is a 32 bit register.

**step 3 cont.**

• These registers are initialized to the

following values in hexadecimal:

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

**Step 4 – Process message in 16-word blocks.**

– Four auxiliary functions that take as input three 32-bit words and produce as output one 32-bit word.

F(X,Y,Z) = XY v not(X) Z

G(X,Y,Z) = XZ v Y not(Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X v not(Z))

**Step 4 – Process message in 16-word**

blocks cont.

– if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z),G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent

and unbiased.

**Step 5 – output**

– The message digest produced as output is A, B, C, D.– That is, output begins with the low-order byteof A, and end with the high-order byte of D.

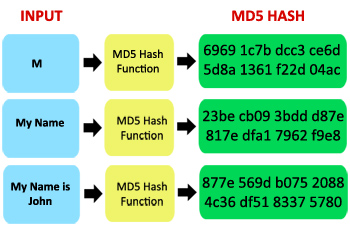
**Summary**

The MD5 algorithm is simple to implement, and provides a “fingerprint” or message digest of a message of arbitrary length.

• The difficulty of coming up with two messages with the same message digest is on the order of 2^64 operations.

**What is MD5 algorithm?**

MD5 which stands for **Message Digest** algorithm **5** is a widely used cryptographic hash function that was invented by Ronald Rivest in 1991. The idea behind this algorithm is to take up a random data (text or [binary](http://en.wikipedia.org/wiki/Binary_file)) as an input and generate a fixed size “hash value” as the output. The input data can be of any size or length, but the output “hash value” size is always fixed. Here is an example of MD5 Hash function at work:



As you can see from the above example, whatever the input size you give, the algorithm generates a fixed size (32 digit hex) MD5 hash. You can even try this on your own using the [MD5 hash generator tool here](http://www.miraclesalad.com/webtools/md5.php).

**Applications of MD5 Hash:**

Before I tell you about how to use MD5, I would like to share one of my recent experience which made me start using MD5 algorithm.

Recently I made some significant changes and updates to my website and as obvious, I generated a complete backup of the site on my server. I downloaded this backup onto my PC and deleted the original one on the server. But after a few days something went wrong and I wanted to restore the backup that I downloaded. When I tried to restore the backup, I was shocked! The backup file that I used to restore was corrupted.

That means, the backup file that I downloaded onto my PC wasn’t exactly the one that was on my server. The reason is that there occurred some data loss during the download process. Yes, this data loss can happen often when a file is downloaded from the Internet. The file can be corrupted due to any of the following reasons:

* Data loss during the download process, due to instability in the Internet connection/server.
* The file can be tampered due to virus infections or
* Due to Hacker attacks.

So, whenever you download any valuable data from the Internet, it is completely necessary that you check the integrity of the downloaded file. That is, you need to ensure that the downloaded file is exactly the same as that of the original one. In this scenario, the MD5 hash can become handy. All you have to do is generate MD5 hash (or MD5 check-sum) for the intended file on your server.

After you download the file onto your PC, again generate MD5 hash for the downloaded file. Compare these two hashes and if they match, that means the file is downloaded perfectly without any data loss.

A MD5 hash is nothing but a **32 digit hexadecimal number** which can be something as follows:

**e4d909c290d0fb1ca068ffaddf22cbd0**

This hash is unique for every file irrespective of its size and type. For example, two different executable files (.exe files) with the same size will not have the same MD5 hash even though they are of same type and size. So MD5 hash can be used to uniquely identify a file. The same thing applies even for messages where each message that was sent and received can be verified using the MD5 hash.

**How to use MD5 Hash to check the Integrity of Files?**

Suppose you have a file called **backup.tar** on your server. Before you download, you need to generate MD5 hash for this file on your server. To do so use the following command: