Message Digest 5 Hashing using Message Passing Interface

Vijay Kumar Sahu1, Aagam Shah2, Manjunath K N3, Neelima B4, Dinesh Acharya U5, Renuka A6  
Department of Computer Science & Engineering  
Manipal Institute of Technology*,* Manipal Academy of Higher EducationManipal, India  
1[vijay.mitmpl2022@manipal.edu](mailto:vijay.mitmpl2022@manipal.edu), 2[aagam.mitmpl2022@manipal.edu](file:///C:\Users\Admin\Downloads\aagam.mitmpl2022@manipal.edu), 3[manjunath.kn@manipal.edu](mailto:manjunath.kn@manipal.edu), 4[neelima.bayyapu@manipal.edu](file:///C:\Users\Admin\Downloads\neelima.bayyapu@manipal.edu), 5[dinesh.acharya@manipal.edu](mailto:dinesh.acharya@manipal.edu), 6[renuka.prabhu@manipal.edu](file:///C:\Users\Admin\Downloads\renuka.prabhu@manipal.edu)

*Abstract*—A trap function, MD5 (message digest), can be used to verify file integrity, typically to prevent accidental corruption. A 128-bit state vector is initialized and modified using shifts and bitwise logical operations on each of the 64 input bytes to create the MD5 hash function. In an MD5 cycle, four non-linear functions are applied 16 times, each time requiring between 7 and 10 processor instructions. Per input byte, this translates to approximately 8–10 machine instructions. In reality, pipelined modern CPUs can compute MD5 at a rate of about 4.5 cycles per byte. Although this is an increase over a non-pipelined system, 4.5 cycles per byte predicts a throughput of only about 600MB/sec in a world of 10Gb networking. for improving hashing speed we will implement MD5 by parallelism with message passing interface(MPI),it can increase it efficiency by at least 2 folds.

Keywords—hash, message passing interface, message digest, parallelism

# Introduction

With a hash value of 128 bits, the MD5 (Message Digest Algorithm 5) is a widely used cryptographic hash function. The MD5 algorithm, which is included in an internet standard (RFC 1321), has been used or developed in a number of security applications. Its main function is to check the authenticity of files or other items. For MD5, the input message's encryption is separated into 512-bit blocks (each with sixteen 32-bit sub-blocks). To produce a 128-bit message digest with four concatenated 32-bit blocks, MD5 goes through a sequence of steps. To compute the message's digest, padding bits are first inserted to make the message's length congruent to 448, modulo 512, and then the length bits. The size of the actual message is indicated by a 64-bit section that is appended. The MD5 algorithm uses a (128-bit) state that is split into four parts and it initialize into 32-bit words (designated as X, Y, Z, and W)[1].

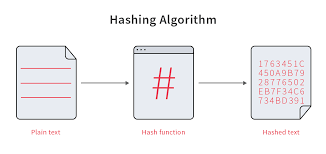


FIG1. Conversion of plain text into hashed text through hashing

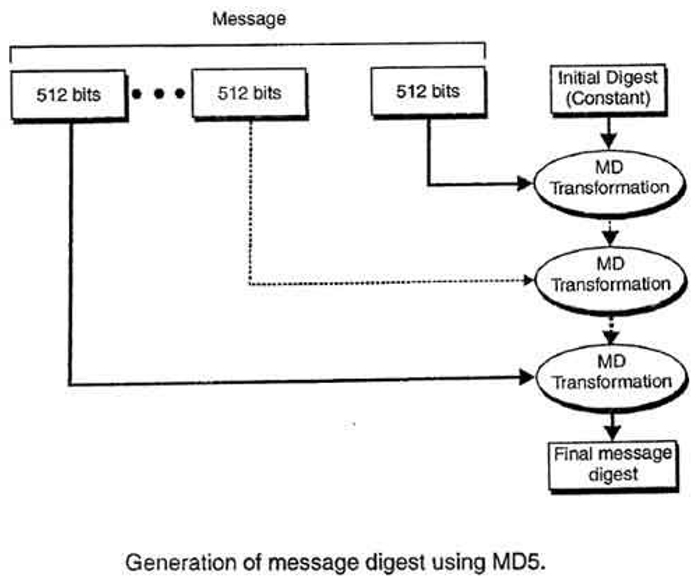


FIG2.Generation of message digest using MD5

The state is modified by applying each 512-bit message block one at a time. A message block is processed in four identical rounds that each consist of 16 identical operations that based on the non-linear function F, modular, addition, and left rotation. After the last round, X, Y, Z, and W are cascaded to create the MD5 output Ease of Use. We learned about MD5 hashing till now for increase its efficiency, we will now execute MD5 with MPI(it is an message passing interface which is used to communicate with other processors in distributed network for attain parallelism in program) to get parallelism.

# Related Work

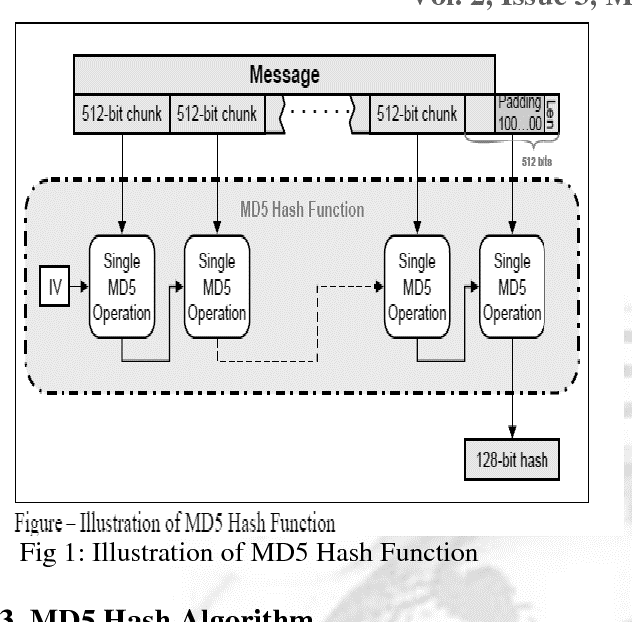
The Bad den Boer and Antoun Boscellars Implement the following C program algorithm. Set a workload that detects about 216 collision in the first twice laps of MD5 compression function for finding collisions in all four round functions. At 33 MHz Average execution time for 80386-based PCs. The program lasts approximately 4 minutes[3].

Since Wang et alfirst .'s practicable MD5 collision differencing in 2004, a lot has been accomplished. The majority of the study has been on how to make it better, but the literature is sparse on how to pick weak input differences to defend against MD5 collision assaults. Wang et al. have proven that multiple hash algorithms are resistant to collisions. The key component of collision attacks on hash functions is the method for identifying message disparities. He has only yet to disclose three additional message differential attacks. One is a difference of six bits, and two is a difference of one bit. due to a 6-bit message discrepancy within a 2-block message, the first MD5 collision was discovered. The best outcome was 230 MD5 procedures after numerous studies greatly raised the computational complexity. No further notice, though. It becomes out that collisions result from differentiation. prior to 2008 [4-5].

The first published single-block MD5 collision was announced by Tao Xie and Dengguo Feng on December 24. (two 64-byte messages with the same MD5 hash)[6]. Multi-block attacks were previously used for collision detection. Xie and Feng did not divulge the new attack strategy "for security considerations." By January 1, 2013, he has issued another $10,000 reward to the first person to discover a 64-byte collision after challenging the crypto community once more.

# Metholdolgy

## Firstly, we will see how MD5 function will do their work with diagram and descriptive representation.



## Fig 3. illustration of MD5 hash function

## Algorithm and method for MD5

1. Algorithm input

* The method accepts input that can be any length in bits. This could be an integer, struct, file, string, etc. It's not necessary for it to be byte-aligned either, but it usually always is. This input will be known as the message. The digest is the result.

1. **Padding:**

* To make the provided message's length conform to "448 mod 512" bits, bits are appended to the end of the message. To put it another way, the message is padded to be 64 bits shorter than the subsequent multiple of 512. The length of the original message is still padded with 512 bits even if it already satisfies this criteria.[7]
* The padding is just a single "1" bit at the end of the message, followed by a sufficient number of "0" bits to meet the previous length requirement. (Note: Strings frequently have null endings. As you will see, this null character is not taken into consideration.) We must now add padding to our message bits.

1. **Appending the Length**

* The length of the message modulus 264 is then added to the message in little endian to bring the overall length to a multiple of 512. This length is equal to the original message's modulus 264 number of bits. Keep careful track of which bytes are placed where; the highest order byte should be the last byte in the message. It's typical to break this number into two 32-bit words. This will bring the total message length to a multiple of 512.
* 0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 The lowest order byte is written first because we are writing in little-endian. Make sure to add the lower order bytes first if you're holding the length in two separate 32-bit words.

1. Processing

* The RSA memo lists four functions that can be used to combine three 32-bit words into a single 32-bit word:

(1)

(2)

(3)

(4)

* These are bitwise operations.
* The bits of a word must also be rotated to the left. In other words, relocate overflow to the right and shift the bits to the left. like when you rotate a bottle and the label loops around. This is how the function is defined:
* rotate left(x, n) = (x << n) | (x >> (32 - n))
* The constants in K and S can be found at the bottom of this section.
* The message is divided into 512-bit blocks. Each block has 16 segments ( 32-bit words). Perform the following for every block:

Md5Finalize(){

int max;

float ub = ceil(16/cluster);

max = ub;

for(unsigned int i = 0; i < max; ++i){

input0[j] = (uint32\_t)(ctx->input[(((max \* rank)+j) \* 4) + 3]) << 24 | (uint32\_t)(ctx->input[(((max \* rank)+j) \* 4) + 2]) << 16 | (uint32\_t)(ctx- >input[(((max \* rank)+j) \* 4) + 1]) << 8 | (uint32\_t)(ctx->input[(((max \* rank)+j) \* 4)]);

}

MPI\_Gather(input0, max, MPI\_UINT32\_T, input, max, MPI\_UINT32\_T, 0, MPI\_COMM\_WORLD);

* Rather of employing control structures, the RSA memo explicitly states each step. The outcome is identical[8].

# Result and Discussions

The proposed algorithm is tested on a 4 core machine with 8 GB of Ram on a Windows operating system. First the sequential version of the algorithm is run, and it utilized, and then by applying (MPI) we parallelize it, And the result of sequential code is shown in (Fig.4) and the running time took 0.766 seconds for 391 input length. Then the parallel version of the proposed algorithm is run on the same input. Monitoring the execution time, we noticed the full utilization of all the cores as shown in the (Fig.5) The overall running time of the algorithm took an average of 0.378 seconds.

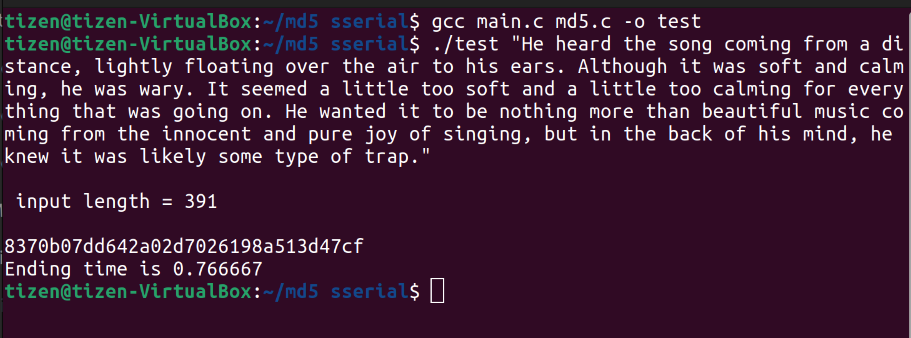


Fig.4 MD5 hashing output using sequential code

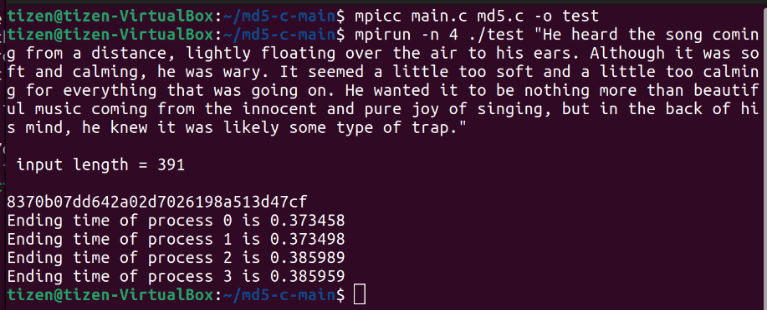


Fig. 5 MD5 hashing output using parallelism(MPI)

# Conclusion and Future Scope

## From result we can conclude that by solving MD5 in parallel approach we get an execution time half than sequential execution time suggest that the work we presented in this paper can be extended in several different directions. To avoid the collision effect for encryption seen in the current implementation, Improving the message-passing method would enable every worker thread to assign work to any thread that asks it. to make it transferable. I believe this is more effective in reducing the amount of time threads spend waiting for work, but can create bottlenecks when multiple threads try to do work and eventually trigger lock or it take more execution time then sequential execution time.so we can remove these collision effect by using (SHA-3).

##### References

1. D. Agrawal, J. R. Rao, P. Rohatgi. "Multi-channel Attacks". In. Cryptographic. Hardware and Embedded Systems (CHES 2003)*Springer-Verlag LNC*, 2003, Volume 2779/2003:2-16.
2. Zhang Shaolan, Xing Guobo, Yang Yixian, “Improvement to MD5 and Security Analysis”. Computer Application, 2009,vol.29(4):947-949
3. J. Dyer, M. Lindemann, R. Perez at al. "Building the IBM 4758 Secure Coprocessor". *IEEE Computer*, 2001, 34(10):57-66.
4. W.X. Zhang, W.N. Chen and J. Zhang, "A dynamic competitive swarm optimizer based-on entropy for large scale optimization [C]", 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI), 2016.
5. P W. Stalling, “*Cryptography and Network Principles and Practices*”, 4th Ed., New Jersey: Prentice Hall, 2005.
6. A. A. Pamungkas, "Implementasi Algoritma Sistem Kriptografi MD5, SHA-1 dan RC4 pada Aplikasi Mobile Internet Berbasis Java," *Journal Penelitian dan Pengembangan Telekomunikasi*, vol. 11, no. 1, June 2006.
7. Xue Jia Lai, "An objective look at MD SHA-1 has been “cracked” [J]", National Information Security Evaluation and Certification, vol. 3, pp. 6-7, 2005.
8. R.L.J.R. Rivest, The MD5 Message-digest Algorithm, vol. 473, no. 10, pp. 492-492, 1992.