ISSN: 3139-0862

An Image to Text Encoding Framework for Image Integrity Verification Using MD5

Md Afsarul Hoque
Assistant Professor, Department of Computer Science and Application, North Bengal St. Xavier's College, NBU
West Bengal, India
ma.hoque.pqc@gmail.com

Abstract —
This paper presents a from scratch implementation and experimental analysis of the Message Digest Five hashing algorithm using C plus plus. The study examines message pre-processing, block processing, nonlinear transformation rounds, and generation of the fixed length digest. In addition to conventional text inputs, a deterministic image to text encoding pipeline is implemented using OpenCV so that image metadata and pixel values can be serialized before hashing. Experiments on modified text and image inputs show that small input changes produce substantially different digest values, demonstrating deterministic behavior and the avalanche effect. For three original and tampered image pairs, the reported digest differences correspond to Hamming distances of 65, 54, and 62 bits out of 128, respectively. The work also discusses appropriate and inappropriate uses of Message Digest Five in view of practical collision attacks. The results support its use only for non security critical integrity checks, teaching, and experimental analysis, while stronger modern hash functions should be used for security sensitive applications.

Keywords—avalanche effect; hash function; image integrity; image hashing; Message Digest; tamper detection; MD5

I. INTRODUCTION

Cryptographic hash functions transform messages of arbitrary length into fixed length digests and are widely used for integrity verification, authentication support, and data processing. A useful hash function is deterministic, computationally efficient, and designed to resist preimage, second preimage, and collision attacks [5], [7]. Message Digest Five, commonly known as MD5, produces a 128 bit digest and became widely adopted because of its simple structure and computational speed [1].

MD5 is no longer considered suitable for security critical cryptographic applications. Practical collision attacks demonstrated that two different inputs can be constructed with the same digest, undermining collision resistance [2], [6]. Consequently, MD5 should not be used for digital signatures, certificate generation, or password storage. Nevertheless, its compact structure remains useful for teaching, implementation studies, and non adversarial file integrity checks where collision resistance is not the primary requirement [3], [7].

This work focuses on two complementary objectives. First, the MD5 algorithm is implemented from scratch in C++ to expose its preprocessing, message scheduling, nonlinear functions, modular operations, and state updates. Second, an image to text encoding pipeline is developed so that both grayscale and color images can be converted into a deterministic textual representation before hashing. The experimental study evaluates the response of the digest to small changes in text and image inputs.

The main contribution is therefore an implementation oriented framework that connects low level MD5 processing with a reproducible image serialization procedure for integrity experiments. The study is intentionally framed as an analysis of non security critical integrity verification rather than as a proposal to replace modern secure hash functions.

II. MATERIALS AND METHODS

A. System Architecture

The implemented system accepts either a text file or an image file. Text input is read as a sequence of bytes and passed directly to the MD5 processing pipeline. Image input is first loaded through OpenCV, after which its dimensions, color mode, file extension, and pixel values are serialized into a standardized text representation. The resulting representation is then hashed using the same MD5 implementation.

The processing pipeline consists of input acquisition, image pre-processing when required, message padding and length encoding, 512 bit block division, four round MD5 transformation, digest accumulation, and final hexadecimal representation. The design allows the same hashing core to be tested on different classes of input while keeping the image conversion stage explicit and reproducible.

ISSN: 3139-0862

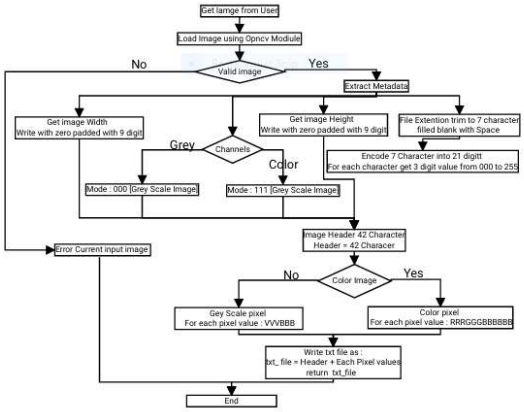

Fig. 1. Image to text encoding process used before MD5 hashing.

B. MD5 Implementation

MD5 processes the padded message in 512 bit blocks. The message is first extended with a one bit followed by zero bits until its length is 448 modulo 512, and the original message length is then appended as a 64 bit little endian value [1]. The internal state contains four 32 bit words, initialized as A equals 67452301, B equals efcdab89, C equals 98badcfe, and D equals 10325476 in hexadecimal notation.

Each 512 bit block is divided into sixteen 32 bit words. The compression function performs 64 operations organized into four rounds. Each round uses one nonlinear function and a defined sequence of message words, constants, and circular left rotations. The four functions are F for conditional selection, G for permutation logic, H for exclusive OR mixing, and I for final diffusion. After all operations for a block, the temporary state is added to the cumulative state. The final four words are serialized in little endian order to form the 128 bit digest [1], [8].

Round	Function	Formula	Purpose
1	F	(x & y) | (~x & z)	Conditional selection of bits
2	G	(x & z) | (y & ~z)	Permutation logic for bit mixing
3	H	x ^ y ^ z	Simple XOR mixing
4	I	y ^ (x | ~z)	Final diffusion and bit manipulation

TABLE I. NONLINEAR FUNCTIONS USED IN THE MD5 ROUNDS.

C. Image to Text Encoding

The image encoding stage was designed to produce a fixed structure before hashing. A 42 character header records the image mode, height, width, and original file extension. The mode field uses 000 for grayscale and 111 for color. Height and width are represented using nine digit zero padded fields, while the extension is represented using seven three digit character codes, producing 21 digits.

Pixel data follow the header in row major order. For grayscale images, each pixel is represented by a brightness bucket followed by a three digit intensity value. For color images, the representation contains a brightness bucket followed by three digit red, green, and blue channel values. This serialization makes the conversion deterministic for the image attributes represented by the pipeline and allows the resulting text stream to be processed by the MD5 core.

D. Experimental Procedure

Text experiments used strings with controlled changes in capitalization, spacing, punctuation, and individual characters. Image experiments used three original images and corresponding modified versions. The modifications included small visual changes and more visible alterations. Each image was converted using the proposed encoding process and then hashed. The resulting digests were compared to determine whether small input changes were reflected in substantially different output values.

In addition to the reported hexadecimal digests, the Hamming distance between each original and tampered 128 bit digest was calculated from the hexadecimal values reported in the source results. This derived measure provides a compact numerical view of the observed avalanche behavior without changing the underlying experiment.

III. RESULTS AND DISCUSSION

A. Text Input Results

The text experiments consistently produced the same digest when the same input was processed repeatedly, confirming the deterministic property of the implementation. At the same time, small modifications produced unrelated looking digest strings. For the reported comparison pairs, the Hamming distances between the original and modified digests were 66, 59, 60, 56, and 60 bits out of 128 for the case, spacing, typo, technology, and paragraph examples, respectively. The average distance for these five comparisons was 60.2 bits, or approximately 47.0 percent of the digest bits.

S.I.	Input Text	MD5 Digest	Remarks
1	university of north bengal	bc90165deee0c348df-f857612492e732	Original lowercase string.
2	University of north bengal	9a8581f8e904219e164 4ec7e80519e3c	Changed 'u' to 'U'.
3	university of north bengal	a9e4334224d8f2fd-c536e1f06836b450	Added an extra space after 'uni-versity'.
4	university of noth bengal	8484d46d99e47176848 deef82b7a370e	Typo: 'north' changed to 'noth'.
5	AABCD EFGH...Technology	55214c1ee9f9491de-b9dda71f74e3e7a	Original text from
6	AABCD EFGH...tech-nology	6b23446e73624e565ae-bcfdb701f5c9b	Changed 'T' in 'Technology' to 't'.
7	Full paragraph text (Appendix A)	cbe9cf07b7caefa82c9d-cafc65a362b4	Original paragraph .
8	Full paragraph, 'and'->'&'	2b2fde8e11a63350b1f-caa3e8b51e59c	Changed a single word in the paragraph.

TABLE II. REPRESENTATIVE TEXT RESULTS AND DIGEST CHANGES.

B. Image Input Results

The image experiments produced a corresponding change in the digest for every original and modified pair. The reported digests for the three pairs are reproduced in Table III. Hamming distances derived from these values were 65, 54, and 62 bits, respectively. The average distance was 60.3 bits, equivalent to approximately 47.1 percent of the 128 digest bits. These values are close to the half bit change expected from a strongly diffusing hash output and provide quantitative support for the qualitative avalanche effect observed in the original project.

ISSN: 3139-0862

S.I.	Image File (Encoded to .txt)	MD5 Digest	Remarks
1	a1.png [Original]	86b8b721e9645499d151068653b2e4a7	Original image (Fig 2(a1)).
2	a1.png [Tampered]	54cbf44ecd-b5b5fe37e5986551ae35c0	Tampered image (Fig 2(a2)).
3	b1.png [Original]	b8fc9860e7317e2-ca77e2bced097bf83	Original image (Fig 2(b1)).
4	b1.png [Tampered]	2144f0c2f469449d-f9426f3654f33ef3	Tampered image (Fig 2(b2)).
5	c1.png [Original]	bc7e75699e-da199078c33acbcf94608b	Original image (Fig 2(c1)).
6	c1.png [Tampered]	a1083a5a858f0813c35033d189251ab8	Tampered image (Fig 2(c2)).

TABLE III. ORIGINAL AND TAMPERED IMAGE DIGESTS WITH DERIVED HAMMING DISTANCE.

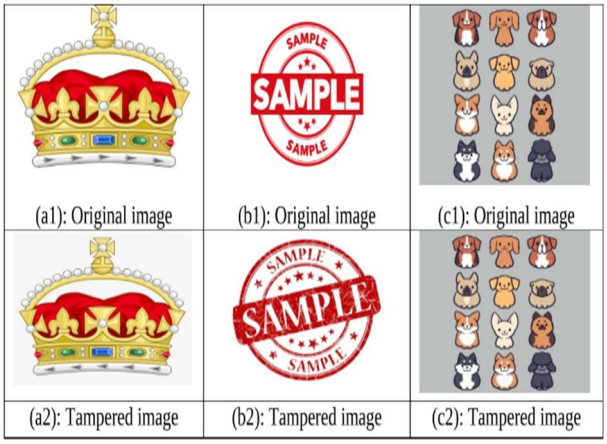

Fig. 2. Examples of original and tampered images used in the experimental evaluation.

C. Comparison With Other Hash Functions

The study places MD5 in context by comparing it with SHA 1 and SHA 256. MD5 and SHA 1 are both unsuitable for collision resistant security applications, whereas SHA 256 provides a substantially larger digest and is part of the SHA 2 family [4]. The original benchmark observations also showed higher processing speed for MD5 than for SHA 1 and SHA 256 on the tested workload. These performance observations are implementation and hardware dependent and should not be interpreted as a security advantage.

Feature	MD5	SHA-1	SHA-256
Output Size	128 bits	160 bits	256 bits
Internal State Size	128 bits (4x32)	160 bits (5x32)	256 bits (8x32)
Block Size	512 bits	512 bits	512 bits
Number of Rounds	64 (4 rounds of 16)	80 (4 rounds of 20)	64
Security Status	Broken (collisions feasible)	Deprecated (collisions practical)	Secure (as of 2025)

TABLE IV. HIGH LEVEL COMPARISON OF MD5, SHA 1, AND SHA 256 [4], [9].

D. Practical Applications and Limitations

The experiments demonstrate that the implementation is sensitive to small changes in the represented input. This behavior can be useful for accidental corruption detection, duplicate identification, and educational demonstrations of hash behavior. For a non adversarial checksum workflow, an MD5 mismatch can indicate that a file representation has changed during storage or transfer.

The same property does not make MD5 suitable for security sensitive applications. Practical collision construction defeats the collision resistance expected from a cryptographic hash [3], [6]. The algorithm should therefore not be used for digital signatures, certificate generation, password storage, or other applications in which an attacker can deliberately manipulate inputs. Modern applications should use a current secure hash construction such as SHA 256 or an appropriate SHA 3 family algorithm.

The image pipeline is also best understood as an experimental representation layer rather than a replacement for hashing the original binary file. Any integrity system should define precisely which image attributes are included in the representation. The present implementation includes the metadata and pixel information specified in its serialization scheme; attributes outside that representation are not independently authenticated. This distinction is important when applying the method to real digital forensics or content management workflows.

IV. CONCLUSION

This paper presented a from scratch C plus plus implementation and experimental analysis of MD5 together with a deterministic image to text encoding pipeline. The text and image experiments confirmed the deterministic behavior of the implementation and demonstrated strong output sensitivity to small input changes. The derived Hamming distances for the reported image pairs were 65, 54, and 62 bits, while the five text comparisons produced distances ranging from 56 to 66 bits.

The principal practical value of the work lies in connecting low level hash implementation with a reproducible representation of image data. At the same time, the analysis reinforces an essential limitation: MD5 is cryptographically broken for collision resistant security applications. Its appropriate role is therefore restricted to non security critical integrity checking, legacy compatibility, controlled experimentation, and education. Future work can extend the same image serialization framework to secure modern hash functions and evaluate performance, representation overhead, and robustness across larger and more diverse image datasets.

REFERENCES

[1] R. L. Rivest, “The MD5 message digest algorithm,” RFC 1321, Internet Engineering Task Force, 1992.
[2] B. Preneel, “The state of cryptographic hash functions,” in Progress in Cryptology INDOCRYPT 2004, vol. 3348, pp. 1–14, Springer, 2004.
[3] V. Klima, “Finding MD5 collisions on a notebook PC using multi-message modification,” IACR Cryptology ePrint Archive, Report 2004/199, 2005.
[4] National Institute of Standards and Technology, “Secure Hash Standard,” FIPS PUB 180-4, U.S. Department of Commerce, 2015.
[5] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.
[6] X. Wang and H. Yu, “How to break MD5 and other hash functions,” in Advances in Cryptology EUROCRYPT 2005, vol. 3494, pp. 19–35, Springer, 2005.
[7] N. Ferguson and B. Schneier, Practical Cryptography. Wiley, 2003.
[8] [8] F. Sagar, “Understanding cryptographic hash functions,” Indiana State University, n.d.

ISSN: 3139-0862

[9] C. K. Koc, ed., Cryptographic Engineering. Springer, 2009.
[10] J. Pieprzyk, T. Hardjono, and J. Seberry, Fundamentals of Computer Security. Springer, 2003.