# Validating Data Integrity in Steganographed Images using Embedded Checksum Technique

Jagan Raj J
MTS, Storage R&D Department,
VMware Software India Pvt Ltd, Kalyani Vista,
165/1, Doraisanipalya, Bengaluru, KA, India - 560076

Prasath S
Asst. Professor, Dept. of Computer Science,
Erode Arts and Science College,
Rangampalyam, Erode, TN, India -638009

## ABSTRACT
In this paper, discuss on validating the data integrity of an image that carries secret information across the network. Validating the data integrity has been always a difficult task on steganographed image files. To discuss a way through which data integrity is verified for possible image tampering by intruders using md5 checksum in self embedded technique.

## Keywords
Image steganography, md5checksum, data integrity, data tampering, message digest, data security

## 1. INTRODUCTION
Pictures are the most common and sophisticated means of conveying or transferring information. A picture is worth a thousand words and they are concisely convey information about various positions, sizes and inter-relationships among or between objects. They portray several information that we can recognize as objects. Human beings are good at deriving any information from these images, because of our innate visual and mental abilities. About 75% of the information received by human are in pictorial form.

The word steganography is of Greek origin and means "covered, or hidden writing"[9]. It is the science of hiding information. On the other hand, cryptography is used to make data under transmission to unreadable by any intruder, the goal of steganography is to hide the secret data from a third party. In steganography, the information can be hidden in any medium such as images, audio files (jpg, png, bmp etc.,), text files (doc, ods, docx etc.,) and video transmissions (mp4, avi, mkv etc.,). When message is hidden in these medium a stego carrier is formed, which is called as stego-image. It will be perceived to be as close as possible to the original carrier or cover image by the human. Steganography and cryptography are closely related in the data security. Cryptography scrambles the given messages so that they cannot be understood. On the other hand, Steganography, will hide the given message in given media file so that there is no knowledge of the existence of secret message in the first place. Steganography includes the hiding of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. [1]

However, today Steganography is very sophisticated than the examples above suggest, allowing a user to hide huge amount of information within image or audio files. These types of steganography are often used in conjunction with cryptographic techniques so that the information is doubly protected; at first it is encrypted and then hidden so that an adversary has to first of all find the information and then decrypt it.

One of the main constraint on steganographic techniques are to verify the data integrity of media file, which carries the original message and the file, which is predominantly image file. There are possibilities, where an intruder can capture the image on transmission and do some manipulations so that the file is altered in such a way that would benefit the intruder's intention and also lead to the possibility of misinterpreting the original information to a wrong one.

The first step in steganography is to pass both the secret message and the cover message ie., the image file, into the encoder. In the encoder, protocols will be implemented to embed the given secret message into the media file. The type of protocol to use will depend on what kind of information you are trying to embed and where you are embedding it in. For example, you can use an image protocol to embed information inside any image file. A key is often needed in sender's end for embedding process. This can be a public or private key, so that you can encode the secret message with your own private key and then the recipient can decode it using his/her public key. When embedding the information in this way, you can reduce the chance of a third party attacker getting hold of the stego object and decoding the same to find out the secret message. In general, the embedding process inserts a mark in an object.

Having passed through the encoder, a stego image will be produced. A stego image is the original cover object with the secret information embedded inside in it. This object should or always look identical to the cover object as otherwise a third party attacker can see embedded information. Having produced the stego image, it will then be sent through some communications channel, such as secure copy, ftp or email to the intended recipient for decoding. The recipient will decode the stego object in order to view the secret information. The decoding process is simply the reverse of encoding process followed. It is the extraction of secret data from a stego image.

In the decoding process, the stego image is fed in to the system. The private or public key can decode the original key that is used during the encoding process is also needed so that the secret information can be decoded in receiver's end. It depends on the encoding technique, where sometimes the original cover object is also needed during the decoding process. Otherwise, there may be no way of understanding or extracting the secret message from the stego image. Once decoding process is completed, the secret message embedded in the stego-image can then be extracted and seen. The generic decoding process again requires object, I'. The result will be either the retrieved secret message from the object or indication of the likelihood of M being present in image I. Different types of robust marking systems use different inputs and outputs. A formula for this process can be:

Cover medium + Secret message = Stego-Image

The typical flow of a steganography process is as mentioned in figure-1.

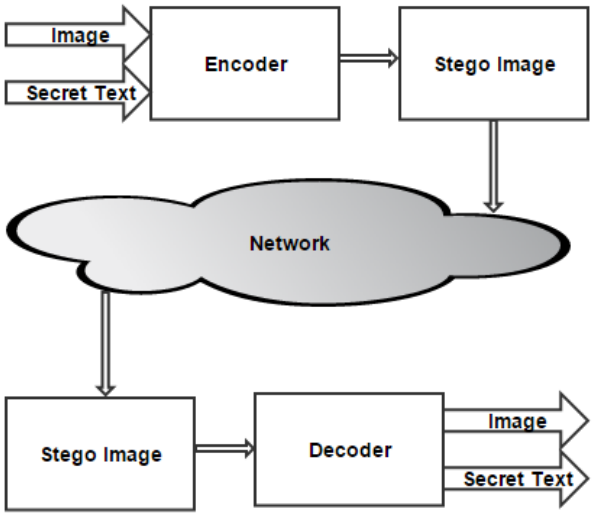


**Fig 1: Steganography Process**

There are more number of algorithms available for image steganography like masking, LSB (Least significant bit method), and filtering etc. Least significant bit method is one of the simplest and popular method for data hiding in steganography. Researchers focus for a long time is on security aspect. A hash function is any algorithm that maps data of variable length to data of a fixed length[3]. The values returned by a hash function are called hash values, hash sums, hash codes, checksums, digests or simply hashes. an example for practical use in data structure is a hash table where the data is stored associatively. Almost all the programming languages has similar data types for programming ease. In this paper our effort to produce a highly secured stego images under human visual system (HVS). In the proposed method, we are using marker technique[8] to insert hash value.

## 2. IMPORTANCE OF DATA INTEGRITY IN STEGANOGRAPHED FILES

Data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle,[2] and it is a significant aspect of any system which is storing, managing, processing, or retrieving data. The term data integrity is broad in scope and have widely different definitions depending on the specific context – even in a single field of computing.

Data integrity is the antonym of data corruption, which is a form of data loss. The overall goal of any data integrity technique is the same: ensure that the data is recorded exactly as intended in any given file or medium and upon later retrieval, ensure the data is same as it was when it was originally stored. In short, data integrity aims to prevent unintentional changes to information by anyway. Data integrity shouldn't be confused with data security, which is the discipline of protecting data from unauthorized access. Data integrity is about technique for making sure that the data you entered are accurate. It is important to double check all the information that you passed on the sender's end should be same in the reciever's end.

To create a digest of the message, we use hash function[3]-[5]. The hash function creates a fixed digest from a variable – length message as shown in below figure

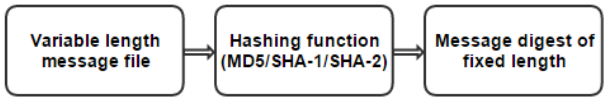


**Fig 2: Message digest generation process**

The two most common hash functions are called MD5(Message Digest 5), SHA-1 (Secure Hash Algorithm 1) and SHA-2 (Secure Hash Algorithm 2). The first one produces a 128-bit digest. The second produces a 160-bit digest. Thr third one produces 224, 256, 384, or 512 bits digest. Hash functions have two properties to guarantee its success.

1. The digest can only be created from message, but not vice-versa.
2. It is one-to-one function and there is little probability that two message will have the same digest.

One practical use in a data structure is called as hash table where the data is stored associatively. Searching for a person's information using name in a list is slow, but the hashed value can be used to assign a reference to the original data and retrieve constant duration. Another use case is in cryptography, the science of safeguarding the data. It is easy to generate hash values from the input data and easy to verify the data matches the hash, but hard to 'imitate' a hash value to hide the malicious data.

## 3. EXISTING APPROACH FOR STEGANOGRAPHY

The existing approach allows the user to embed their secret message in images in such a way that it is invisible and doesn't degrade or affect the quality of the original image[6],[7] to the normal human eye.

1. *Input Image-* An input interface (see figure.1) is provided so that a user can input a (gif, bmp, jpg, or tiff etc.) image in which the user wants to hide their personal data for privacy purposes.
2. *Input Secret Text-* Input the text file containing the Secret Text, which the user wants to code in to the image. The input text file is read by our system (see figure.1).
3. *Coding Data in Image-* For coding text data in the image, several encoding techniques like LSB, marker techniques are used in *steganography*.
4. *Decoding Data from Image-* For decoding secret data in the image. The corresponding algorithms are used and generated characters are concatenated to form a complete secret message

## 4. PROPOSED METHODOLOGY

In this method I have proposed two steps for Steganography

1. Encoding,
2. Decoding
3. Integrity verification

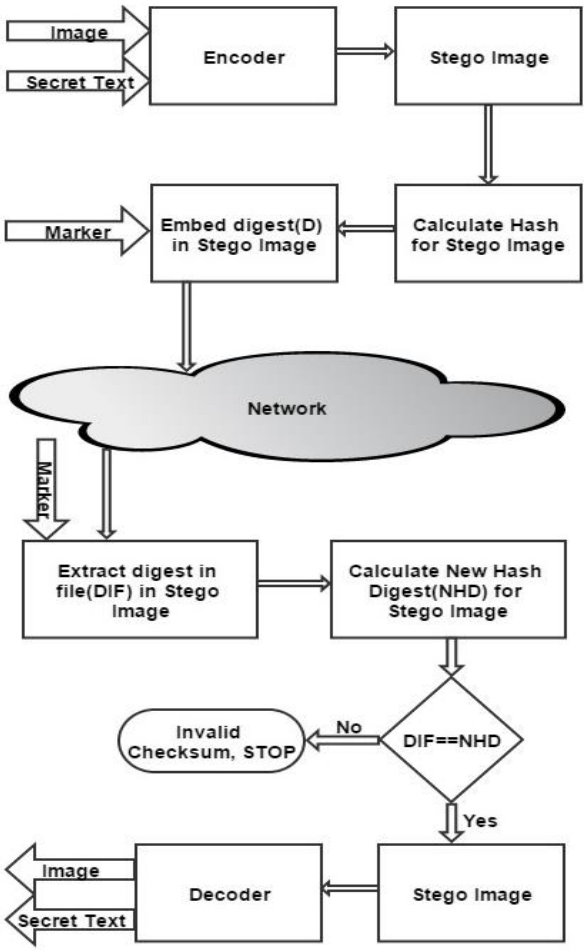


**Fig 3: Proposed steganography process with data Integrity check**

## 4.1 Encoding Algorithm

Proposed Algorithm for Encoding Data in Image:

Step-1: Read the RGB image of any size.

Step-2: Read the secret text (ST) and store the data in the given Image file (I) using any steganographic algorithm(SA)

I=addTextUsingSteg(I,ST,SA)

Step-3: Find hash digest (D) for file (I) from any one of the hash algorithms(MD5/SHA1/SHA2)

D=hash(I)

Step-4: Specify start of marker(SOM) at the end of image file, append the digest (D) calculated in the image file(I) and then specify end of marker(EOM) in the image

I=I+(SOM+D+EOM)

Step-5: Now, the output image (I) containing coded data and hash is ready for transit.

## 4.2 Decoding Algorithm

Proposed Algorithm for Decoding Data in Image:

Step-1: Read the RGB image (I) at the receiving end.

Step-2: Extract the secret text (ST) and store it using steganographic algorithm used in encoding

ST=extractTextUsingSteg(I, SA)

## 4.3 Integrity verification Algorithm

Proposed Algorithm for Data Integrity verification for secret text in Image file:

Step-1: Traverse the image file and fing start of marker(SOM) at the end of image file, read the digest in file (DIF) stored from image file (I)

Step-2: Truncate the hash section(SOM+D+EOM) from the image file(I)

I=I-(SOM+D+EOM)

Step-3: Find new hash digest (NHD) for new image file (I) using the same hash algorithm used for encoding

NHD=hash(I)

Step-4: If digest in file(DIF) and new hash digest(NHD) are equal, the secret text(ST) obtained is valid else invalid

# 5. SAMPLE RESULTS

In our proposed methodology have taken three different color images 256x256-Person, 950x534-Peacock and 1024x768-Nelumno_nucifera of different sizes. Simulation results are performed in Microsoft's File Checksum Integrity Verifier version 2.05 version and Virtual Steganographic Laboratory-1.1 version

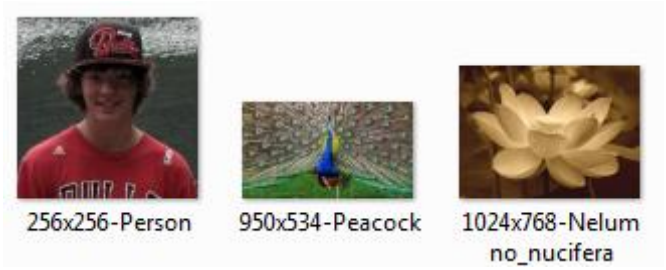


**Figure-4: Original Images**


**Figure-5: Encoded Images with text data and checksum**

The comparison of results with data transfer without corruption/intruder's intervention are shown in Table-I. The files which have changed during transmission by size and content are described in Table-II and Table-III respectively with their corresponding checksums.

**Table I - Comparison of checksum for the steganographed image, which transferred without any corruption**

| **Image(256x256-Person) with 256x256 dimension and undisturbed during the file transit** | | |
|---|---|---|
| **Image stage** | **Checksum value(md5) (D)** | **Secret Text (ST)** |
| After Encoding | 3845e97369852af8c7741ba0ed5ae539 | This is a secret text, which is hidden in a image file using steganography and having embedded checksum in it |

| | | |
|---|---|---|
| After Decoding | 3845e97369852af 8c7741ba0ed5ae5 39 | This is a secret text, which is hidden in a image file using steganography and having embedded checksum in it |

**Table II - Comparison of checksum for the steganographed image, which transferred with corruption because of image resize**

| **Image(950x534-Peacock) with 950x534 dimension and disturbed during the file transit by resizing the image dimension to 475x267** | | |
|---|---|---|
| **Image stage** | **Checksum value(md5) (D)** | **Secret Text (ST)** |
| After Encoding | c965b6c78382dcaf81 037fd28fd6fe37 | This is a secret text, which is hidden in a image file using steganography and having embedded checksum in it |
| After Decoding | 67cb60d47fbec5b947 43ca7706477892 | Secret text got corrupted |

**Table III - Comparison of checksum for the steganographed image, which transferred with corruption because of image color change**

| **Image(1024x768-Nelumno_nucifera) with 1024x768 dimension and disturbed during the file transit by changing the image to black and white** | | |
|---|---|---|
| **Image stage** | **Checksum value(md5) (D)** | **Secret Text (ST)** |
| After Encoding | e64d69492b460cd2 5dbb42f970409f23 | This is a secret text, which is hidden in a image file using steganography and having embedded checksum in it |
| After Decoding | e1c2e6f45c57978c8 6a78df764295972 | Secret text got corrupted |

**Table IV - Qualitative Comparison of proposed methodology with Discrete Cosine Transform(DCT) for below given six parameters**

| Parameters | DCT methodology (Existing) | ECT Methodology (Proposed) |
|---|---|---|
| Digest Inclusion on Stegno files | No | Yes |
| Capability to identify MITM(Man In The Middle) attack | No | Yes |
| IPv4 Header Checksum check | Yes | Yes |
| Digest size used(md5) | 0 bit | 128 bits |
| Robustness | Less data loss | No Data loss |
| Data Integrity Check at receiving end | No | Yes |

# 6. CONCLUSION AND FUTUR WORK

The proposed method is good for security aspect because if any unauthorized users or intruders tampered the image in any aspect and if the data is lost, this can be verified by the embedded checksum. Through this method, we are ensuring the image & the text are intact and the message can't be mis-interpreted in the receiving end. Even though if intruder found the algorithm used for steganography in the stego image by steganalysis and changed the content of the secret text and if the same is received at the receiver's end the tampering can be found by comparing the checksum.

In this methodology, rather than having constant checksum using marker method, we can have the checksum in reserved bytes of file header or the location information from where the checksum starts and its offset.

# 7. REFERENCES

[1] Compression Algorithms for Real Programmers, Wayner Peter. Publisher: Morgan Kaufmann, 11 Oct 1999 ISBN-10: 0127887741

[2] Boritz, J. "IS Practitioners' Views on Core Concepts of Information Integrity". International Journal of Accounting Information Systems. Elsevier. Retrieved 12 August 2011.

[3] From Wikipedia, "Hash function," Jan 2014, URL: http://en.wikipedia.org/wiki/Hash_function

[4] Behrouz A. Forouzan,"Data communication and networking," Tata McGraw-Hill Publication, 2nd ed., 2003, pp.799-800.

[5] William Stallings," Cryptographic and network security," Pearson Prentice Hall Publication, 4th ed., 2006, pp.320-375

[6] Holub V, Fridrich J: Digital image steganography using universal distortion. In 1st ACM Information Hiding and Multimedia Security Workshop. Montpellier; 17–19 June 2013.

[7] W. Bender, "Techniques for Data Hiding," IBM Systems Journal, Vol. 35, no. 7, Pgs 313-336, 1996

[8] Wang Qian, et .al,"Steganography and Steganalysis based on digital image," IEEE 4th International Conference on Image and Signal Processing, Shanghai, 15-17 Oct. 2011, pp.252-255.

[9] Adity Sharma, Anoo Agarwal and Vinay Kumar, " A simple technique for steganography", arXiv:1307.8385v1 [cs.MM] 31 Jul-2013.

[10] Vojtěch Holub, Jessica Fridrich, Tomáš Denemark "Universal distortion function for steganography in an arbitrary domain", EURASIP Journal on Information Security, January 2014.

[11] Denemark T, Fridrich J, Holub V: Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014. Edited by: Alattar A, Memon ND, Heitzenrater CD. San Francisco; 2–6 February 2014