JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

Recursive Equation Approach of Information Hiding for Authentication of Digital Data

Ruchika Sharma
Assistant Professor, Jagan Nath Institute Of Management Studies, IMS, New Delhi-110085
E-mail: msgruchi2@gmail.com

Dr. Vinay Kumar
Ex Scientist, GOI and Ex Dean & Professor, VIPS, GGSIPU, Delhi, India
E-mail: vinay5861@gmail.com

ABSTRACT
As the Information communication technology advances, it has become crucial to ensure the originality of digital content. This paper focuses on using recursive equation approach of steganography to ensure authenticity of digital content. This is achieved by embedding hash value of the digital data in the same digital file. In this paper, we have taken 24 bit BMP files to implement the concept. We have used recursive equation of any order as key to find pixel location in the BMP file to embed the 128 bit MD5 value of the BMP file before its transmission. At the receiving end the MD5 is extracted from the zero bytes of BMP file and the original bits in extra bytes are restored before computing the MD5 value of the received and restored BMP. The two MD5 values are compared. Equality of the value authenticates the originality of BMP file.

Keywords- steganography, recursive equation, MD5, hash value, authenticity.

1. INTRODUCTION

With the commencement of computer and its ever expanding application in different areas of life, the concern of information security has become gradually important. Different methods have been adopted to provide security to digital content such as cryptography and steganography etc. Besides implementing stegnographic approach, it has also become important to ensure the authenticity of digital content. No unauthorized user should be able to hamper originality of digital content[18]. For secure communication, not only security of information is important but also source from where the information is originated is also required [1]. With confidentiality, authentication of original message is also essential. Different methods have been used to verify the originality of the digital information. This paper proposes to use recursive equation method of steganography to hide MD5 Hash in 24 bit BMP file format[13] and then verifying the authenticity of BMP file after obtaining the hash value using MD5 algorithm.

1.1 Steganography

Information hiding is the process of hiding the information in any media e.g text, images audio etc. Steganography means covered writing [2].Steganography is used for hiding information in digital format to fleece the existence of hidden information [3]. Steganography is an art used to maintain confidentiality of information [4]. The purpose of steganography is to hide the fact that communication is taking place. For establishing stegnographic communication, an embedding and extraction algorithm with optionally a key is needed[5]. Steganography may be classified as pure, Symmetric and asymmetric stegnography[18]. In pure steganography, no secret key is shared between sender and receiver whereas in symmetric and asymmetric steganography, a key is shared between two parties [6].In the proposed system pure steganography is used. Most important factor in steganography is the type of medium used in steganography .Highly recommended medium for steganography is image because we may get different randomized pixel locations in image file to hide information. Images can be used as potential medium for hiding information because of its attractiveness. Because of this advantage, In this paper we are using 24 bit BMP file as a medium to verify the originality of digital content by hiding MD5 hash value in 24-bit bmp file[16].

1.2 Bitmap Format

Bmp files are simple and large in size and due to this advantage, these files have got wide acceptance in windows programs. The Microsoft Windows bitmap file format can support a lot of different options, including compression, different bit depths, etc [7].Bitmap files can be stored in different bit depths which determines number of bits used in color palette for each pixel. One bit bmp file format use only one colour, 4 bit bmp uses 16 colours, 8 bit uses 256 colours and 24 bit bmp uses

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

2^{24} bit colour combinations for each pixel[20]. We are using BMP file format to explain our approach.

The bitmap file formats can use little endian or big endian format to store values in file. In little endian format, the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address. For example, a 4 byte Long integer will be written as

Byte3 Byte2 Byte1 Byte0

Whereas in Big endian format, the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address. For example, a 4 byte long integer would be written as:

Byte0 Byte1 Byte2 Byte3

The .bmp file format requires that the digital content in a scan line (horizontal line) in the image should be aligned on a 4-byte boundary, so every line must end with 4 byte boundary. This means that width of an image must be a multiple of 4. If the width of bmp image file is not in multiple of 4 then extra bytes are padded. These Extra bytes are also termed as zero bytes.

In 24 bit bmp, 3 bytes are used to represent a single pixel. So to align a 24 bit bmp along a scan line, (width *3) must be a multiple of 4. For e.g. If we consider a 24 bit bmp image of size 70*80.The total size of bmp file is 5600 pixels which has 70 columns(width) and 80 rows(height).In a single scan line, 70*3 =210 bytes need to be stored but as 210 is not a multiple of 4 ,padding is required. For Padding, we require extra bytes and it can be calculated by the following formula:

Extra bytes = (4 - (3 * width) % 4) % 4

In the above example for storing 210 bytes, 2 extra bytes are required. So in each single line, 212 bytes are required, which includes 210 actual data along with 2 extra or zero bytes. This paper proposes to use 24 bit bmp and a recursive equation approach for hiding message digest to verify the authenticity of the source of file by using extra bytes of bmp file.

1.3 Message Digest

Message digest is a type of cryptographic hash value which can be used to ensure integrity and authenticity of a message. In the context of message verification, a hash function takes input message of any length and delivers a fixed size output. The output is referred as hash or the message digest. The message digest depends totally on bits in input message. The benefit of creating message digest is that a minor change in the input message can totally change its message digest. If during transmission an intruder tries to make any alteration in input message then its message digest would also be changes and will not match the original message digest. To obtain message digest different algorithms like MD5 and SHA-1[8] are used[14, 15]. The obtained message digest from hash function varies in length from 128 bits to 160 bits depending upon the algorithm used [9]. Practically all algorithms divide the long message into equal size blocks and then message is processed block by block in an iterative manner to generate its digest.

2. Recursive Equation approach

A sequence is an enumerated collection of objects. Sequence can be explicit or recursive. A **recursive definition** consists of two parts. First is recurrence relation and second is an initial condition[17].Recurrence relation is an equation that uses a formula to generate next term in the sequence based on previous term and to generate first term, initial condition is required[10]. Recursive equation always have initial condition to get first term and then it repeats itself to generate next terms[21]. In computer science, Recursive methods are used to repeat itself to generate the output[11]. In the proposed paper, we are using recursive equation to generate recursive pixel positions to hide MD5 hash code in .bmp file format to verify the originality of digital content

3. Conceptual Approach

Although MD5 hash codes are difficult to crack for an intruder, to provide one more layer of security we are using recursive approach to hide message digest in cover file[12]. This paper focuses on using 24 bit bmp file to hide hash code to verify the originality of the source file. The MD5 hash code is hidden using recursive equation approach. The recursive equation produces different randomized locations in .bmp file to hide hash code in extra bytes of bmp file[11]. As this paper proposes to hide the message digest in extra bytes, the quality of bmp image will not be degraded. To hide message digest, we are only using the extra bytes which are used to pad the image to provide 4 byte boundary. The concept of paper is based on message digest, recursive equation and 24 bit bmp file.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

The concept is explained in below mentioned steps:

3.1 The 128 bit message digest generated by MD5 algorithm will be hidden in 24 bit bmp file.

3.2 To hide this message digest, extra bytes of bmp file are used. Initially all 16 bits in extra byte will be initialized with number 0 to signify that bit places are vacant and can be used to store message digest

3.3 Recursive equation generates the bit position to find the location in bmp file.. The generated bit position is then moved to first bit of 2 extra bytes at the end of each scan line

3.4 Different bit locations will be generated by recursive equation. First 2 bit positions generated by recursive equation will be used for hiding 2 bits of message digest. The generated bit positions by recursive equation will be moved to first 2 bit position in extra bytes.

3.5 The algorithm will check for free space in 2 extra bytes i.e. 16 bits. If first 2 bits of extra byte have 0 values then 2 bits from 128 bit message digest will be stored in first 2 bits of extra byte and then for storing next 2 bits of message digest, we will check if the next 2 bits from remaining 14 bits of extra byte have 0 value then at these locations we will hide next 2 bits from remaining 126 bits of message digest and this way we will hide 128 bits of message digest in extra bytes

3.6 Masking is used to hide bits from message digest in extra bytes

Figure 1 shows the process of checking authenticity of digital content

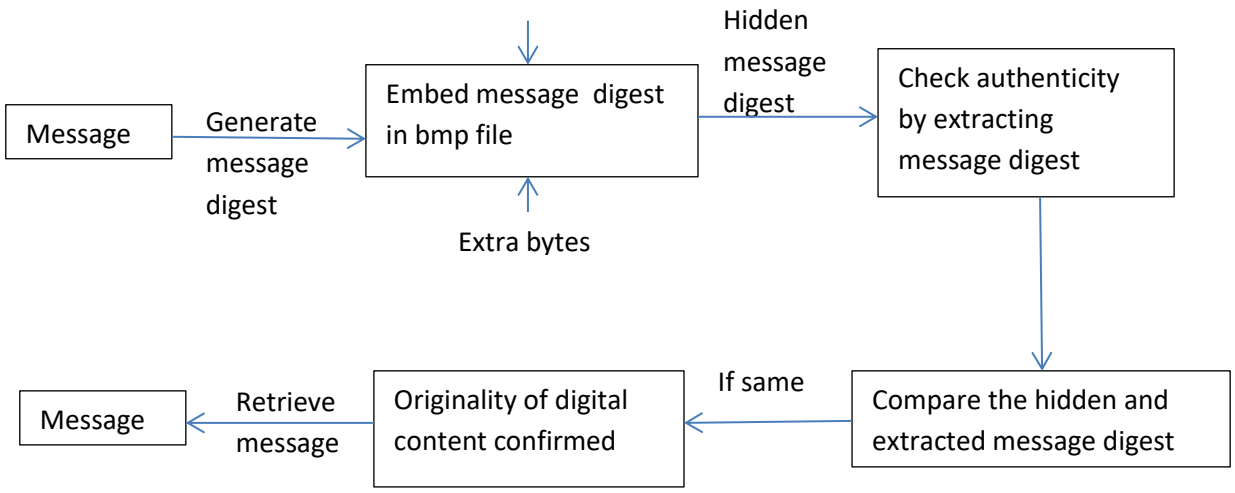

Figure 1

4. IMPLEMENTATION

4.1 Consider a hexadecimal message digest as d50ea9e8881e8b2d e2dd75074a230533 to be hidden in 24 bit bmp file. The generated message digest of 128 bits will be hidden in extra bytes of bmp file. 2 bits from 128 bit message digest will be stored in extra bytes in each scan line. So we need minimum 64 positions to hide 128 bit message digest.

4.2 Let the size of bmp file is 70*80. As each scan line requires a multiple of 4, the width of the image need to be padded accordingly. So to store 210 bytes in each scan line, we need 2 extra bytes to make it 212 bytes which is a multiple of 4.The generated message digest will be stored in these 2 extra bytes in each line.

4.3 Total size of image = 70 *80* 3= 16800 bytes but as we know in each single line the padding is added. In our example we have considered padding of 2 bytes to make it a multiple of 4 in single row.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

So total size after padding in all 80 rows = 16800 + (2*80)
= 16800 + 160
= 16960 bytes

Following table explains the process of hiding message digest in nearest zero byte.

Table 1

	Actual Data Bytes	Zero Bytes	
1	16 53 175 210		212
213	422		424
425	578 634		636
637	846		848
849	1058		1060
1061	1270		1272
1273	1482		1484
1485	1694		1696
1697	1906		1908
1909	1909 2118		2120
2121	2330		2332
2333	2542		2544
2545	2754		2756
2757	2966		2968
2969	3178		3180
3181	3390		3392
3393	3602		3604
3605	3814		3816
3817	3864 4026		4028
4029	4238		4240
4241	4450		4452
4453	4662		4664
4665	4874		4876
4877	5086		5088
5089	5298		5300
5301	5510		5512
5513	5722		5724
5723	5934		5936

4.4

Using recursive equation, generate the pixel locations to hide message digest
Let the equation be

$t_n=at_{n-1}+t_{n-2}$

Constant value of a=7
Initial value of t_0=1
Initial value of t_1=5

Also take a single dimensional array, let a be an array. The array will hold the locations generated by a recursive equation. Initially all places in array will have zero value which implies free space is available.The purpose of using array is to store the locations of hidden message digest

S0, $t_2=3*t_1+t0$
$= 3*5 + 1= 16$

First place generated by recursive equation is 16 so 2 bits from message digest will be stored in nearest zero byte. In this case as 16^{th} byte position is nearest to zero byte in first scan line so 2 bits from message digest will be stored in 211^{th} byte location starting from LSB bit of 211^{th} byte.
we have considered message digest in hexadecimal form
message digest = d50ea9e8881e8b2d e2dd75074a230533

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

The first two bits to be hidden are 11 which is taken from the hexadecimal value of d whose binary equivalent is 1101. Lets consider a character CH=00000000 (all clear bits of 211th byte)

CH1=00000001 (first bit to be hidden in 211th byte)

We will perform OR operation between CH and CH1

CH || CH1 which is also called masking.

CH= 00000000

CH1=00000001

OR 00000001

So after performing OR operation, 211th byte will have 00000001 bits. Now to hide another bit i.e. 1 again, perform OR operation again . Consider ORing output as new 211th byte so new ch = 00000001 and to hide another bit 1 in 211th byte, new ch1=00000010 is considered.

CH = 00000001

CH1= 00000010

OR 00000011

By following above procedure, we have successfully hidden 2 bits from 1101 which is equivalent of 'd' in 211th extra byte. Now to hide next 2 bits i.e. 01, next location need to be generated using recursive equation.

$t_3 = 3*t_2 + t_1 = 3*16 + 5 = 53$

From remaining 126 bits of message digest, next 2 bits i.e. 01 will be hidden in the extra byte nearest to 53rd location generated by recursive equation. So nearest extra byte is again 211th byte. As we have already hidden 2 bits in 211th byte and resultant 211th byte after hiding first 2 bits is 00000011.

To hide third bit i.e. 0, nothing need to be done as in 211th byte, it is already 0 at third LSB location so no need to make any change in current 211th byte. To hide fourth bit i.e., 1, again we have to perform OR operation.

CH =00000011 (211th byte)

CH1=00001000 (Bit presentation to hide 1 at fourth bit position)

OR 00001011

The 211th byte after hiding four bits looks like 00001011. The first 4 bits which are represented as 'd' in message digest in completely hidden. To hide next 2 bits from remaining 124 bits message digest, binary equivalent of 5 is considered which is 0101. So now 01 will be hidden at new bit position generated by recursive equation.

$t_4 = 3 * t_3 + t_2 = 3 * 53 + 16 = 175$

The equation generated 175th location which is again near to 211th byte. As already 4 bits of 211th byte have been used to hide 4 bits of message digest, next 2 bits i.e. 01 from message digest will be hidden at 5th and 6th bit position of 211th byte. To hide '0' , no need to make any change in current 211th byte as 0 is already stored at 5th bit position of 211th byte. Next bit to be hidden is 1 at 6th bit position of 211th byte. Again we will perform masking to hide the bit.

CH = 00001011 (211th byte)

CH1= 00100000 (Bit presentation to hide 1 at sixth bit position)

OR = 00101011

The 211th byte after hiding six bits looks like 00101011. Till now we have hidden 6 bits from message digest. New bit position needs to be generated to hide next 2 bits i.e., 01

Next location generated by recursive equation is :

$t_5 = 3 * t_4 + t_3 = 3 * 175 + 53 = 578$

t_5 has generated 578th byte location and its nearest extra byte is 635. Now next 2 bits i.e., 01 will be hidden in 635th extra byte. Initially all bits of 635th byte will be clear i.e., 00000000. To hide '0' which is seventh bit of message digest, no change is needed because '0' is already stored there. To hide seventh bit, 635th byte need to be changed by performing OR operation with new mask bits.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

CH = 00000000 (635th byte)
CH1 = 00000010 (masking bits to hide 1 at second LSB position of 635th byte location)

OR 00000010

So 2 least significant bits from 635th byte are occupied as we have hidden seventh and eighth bit form message digest. Now from message digest d50ea9e8881e8b2d e2dd75074a230533, we have already hidden 8 bits which means d5 is completely hidden. To hide next bits, consider hexadecimal 0 whose binary equivalent is 0000. From right side, we will hide 2 bits at next location.

The next location generated by equation is as follows:

$t_6 = 3 * t_5 + t_4 = 3 * 578 + 175 = 1909$

The equation generated 1909th byte location which is near to 2119th zero byte. So next 2 bits from message digest i.e., 00 will be hidden in 2 least significant bits of 2119th byte. As per the algorithm, no masking bits are needed as 2 least significant bits are already having 0 value.
The same process is followed to hide complete 128 bit message digest in zero bytes.

5 CONCLUSIONS

The algorithm that has been proposed provides a solution to hide message digest in extra bytes of BMP file format. Hiding secret data in extra bytes has one very crucial advantage of not hampering the quality of image at all. Because the data is concealed in 24 bit BMP, it does not harm the quality of image. As the proposed algorithm hides the data only in extra bytes of image, the intruder will not be able to even detect the presence of hidden data because the actual bits are not disturbed at all to hide data.

References

[1] Chopra, D. et al. 2012. Lsb Based Digital Image Watermarking For Gray Scale Image. Journal of Computer Engineering. 6, 1 (2012), 36–41.
[2] F.A.P Petitcolas, R.J. Anderson and M.G. Kuhn ; “Information Hiding a Survey”, Proceedings of the IEEE, vol.-87, issue 7, pp. 1062-1078, 1999.
[3] A.A.Zaidan, B.B.Zaidan, Anas Majeed, "High Securing Cover-File of Hidden Data Using Statistical Technique and AES Encryption Algorithm", World Academy of Science Engineering and Technology(WASET), Vol.54, ISSN: 2070-3724, P.P 468-479.
[4] H. L. Hussein, "Hiding Data in Color Image Using Least Significant Bits of Blue Sector," Ibn Al-Haitham J. for Pure & Appl. Sci., vol. 31, no. 2, pp. 193-198, 2018.
[5] Cole, E., (2003). Hiding in Plain Sight: Steganography and the Art of Covert Communication. USA: Wiley Publishing.
[6] Beenish Mehboob and Rashid Aziz Faruqui, “A steganography Impleme -ntation”, IEEE-Symposium on Biometrics & Security technologies, ISBAST’ 08, 23-24, April, 2008 Islamabad.
[7] “Image file formats." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 1 February 2014. Web. 25 January 2014. http://en.wikipedia.org/wiki/Image_file_formats
[8] Bellovin, S.M. and Rescorla, E. K. (2005, October). Deploying a new hash algorithm. NIST Hash Function Workshop.
[9] Rivest, R. (1992, April). The MD5 message-digest algorithm. RFC 1321, IETF
[10] Kumar, V., (2002). Discrete Mathematics. New Delhi, India: BPB Publication.
[11] Sharma. Ruchika, Kumar. Vinay, (2020),“Information Hiding using Linear Recursion” in IJSER Volume 11, Issue6, ISSN: 2229-5518.
[12] Kumar, V. and Muttoo, S. K. (2009). A data structure for graph to facilitate hiding information in a graph’s segments – a graph theoretic approach to steganography. Int. J. Communication Networks and Distributed Systems, 3(3), 268–282.
[13] Schaefer Gerald, Stich Michal, “UCID - An Uncompressed Colour Image Database.”, School of Computing and Mathematics. Nottingham Trent University, UK (2003).
[14] Wang, X., Feng, D., Lai, X., Yu, H. (2004) ‘Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.’, IACR Cryptology ePrint Archive, 2004, 199.
[15 Gueron, S., Johnson, S., Walker, J. (2011) ‘SHA-512/256’, in Information Technology: New Generations (ITNG), 2011 Eighth International Conference on, IEEE, 354–358.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 2, 2022, p. 813 - 819
https://publishoa.com
ISSN: 1309-3452

[16] Rogaway, P., Shrimpton, T. (2004) 'Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance', in Fast Software Encryption, Springer, 371–388
[17] Nahi, N.E.; Franco, C., "Recursive Image Enhancement--Vector Processing," Communications, IEEE Transactions on, vol.21, no.4, pp.305, 311, Apr 1973 doi: 10.1109/TCOM.1973.1091662 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1091662&isnumber=23
[18] Adv. Rohas Nagpal, Cyber Terrorism in the context of Globalization, UGC sponsored National Seminar on "Globalization and Human Rights", Mumbai, India 7- 8September 2002.
[19] Bender W., D. Gruhl, N. Morimoto, A. Lu, 1996, Techniques for Data Hiding, IBM Systems Journal 35 (3&4):313-336.
[20] Wayne Brown and Barr. J Shepherd, "Graphics File Formats: Reference And Guide", Manning Publications, Greenwich, Conn, (1995).
[21] Sharma Ruchika, Kumar Vinay, "Implementation of Steganogrphy Using Recursive Equation Approach", IJCMS Vol. 4, Special Issue, (May 2015).