International Journal of Engineering &Technology, 7 (4) (2018) 7079-7085

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.22273
Research paper

A novel LSB steganographic method using partition and matching algorithms

Sukhjit Singh [1] *, Dr. V.K Jain [2], Dr. Amanpreet Singh [3]

[1] Research Scholar, Sant Longowal Institute of Engg. & Tech. Longowal, Punjab, India
[2] Professor Sant Longowal Institute of Engg. & Tech. Longowal, Punjab, India
[3] Depty Controller IKG Punjab Technical University, Kapurthala, Punjab, India
**Corresponding author E-mail: sukhjit_singh@rediffmail.com*

Abstract

There is always a trade-off between embedding efficiency and hiding capacity of secret information. This paper presents a new steganographic technique that improves the embedding efficiency and hiding capacity. For hiding information we extract LSB bits from 24-bit bmp color image. The LSB bits are joined to form an array of bits. An array is divided into N partitions depending upon the size of secret information to hide. The secret information is also divided into N partitions. Each data part finds the best possible matching location within each LSB array partition. The maximum bit matching depends upon the size of data partition, size of LSB array partition and the number of partitions used. The novelty of the proposed method is that the embedding capacity can be increased to desired level with lesser effect on embedding efficiency while maintaining security of embedded data. The paper presents efficient algorithms for bit manipulation that decreases the distortion and increases the embedding efficiency. The proposed technique shows an improvement over the existing LSB embedding methods.

Keywords: *Array Partitioning; Matching; LSB Array; Algorithm; Steganographic System.*

1. Introduction

Information security is the main issue to deal with today. We hide the secret data in image, audio or video files. These carrier files are transmitted through different public channels without any indication that information is hidden in these files. There is another field called steganalysis that deals with how to detect a hidden message in any media such as these carrier files. The main aim of steganalysis is to detect whether a given media contains secret information. To determine the abnormal behavior of an image or any multimedia file different techniques can be used that help in detecting the existence of secret data. Some of the anomalies are undetermined and do not have noticeable visible effects.

The main goal of steganography is to communicate in a secure and undetectable manner without any suspicion. If a steganography method causes someone to reveal the secret or existence of message in the carrier then the method become useless. Steganography provides high level of secrecy and security by combining with cryptography. There are many examples of using steganography in the history. In world war 2, invisible ink was used to write on a piece of paper. The paper appeared blank to the average person. Instead of ink, liquids such as mil, vinegar and fruit juices were used. When such substances are heated they darken and become visible to the human eye. The Greece used to select messengers and shave their head, they would then write a message on their head. After the message had been written, the hair was allowed to grow back. After the hair grew back the messenger was sent to deliver the message. The recipient would shave off the messenger's hair to see the secret message. In image steganalysis, the process is opposite to that of image steganography. Image steganalysis is a way to detect presence of secret message in a carrier. It may further estimate the potentially hidden information from a given image. A 24 bit color image contains three bytes for each pixel in the form of red, green and blue color components. Inserting one bit in each byte stores three bits in a pixel. Thus in an image of 400x300 pixel image contains a 360000 bits (45000 bytes) of secret information. In case the secret information is large then we can use higher bits in a byte to hide data. A run length matching scheme is proposed [1] in which run length table is used. The table is constructed from the secret data as well as the cover data. It compares the RLM and RRLM methods. The disadvantages of this method are that the table generated requires more space as compared to proposed method. Kekre et al. [2] suggested how to embed variable number of secret message bits in different cover image pixels. The method distributes the image pixel values into four different ranges, R1, R2, R3 and R4. Then 'i' number of bits are embedded in range R_i for i=0 to 3. The use of adaptive bits increases the security of algorithm. The disadvantages of this method is that embedding efficiency is not achieved. Mathkour et al. [3] used a method based on spiral LSB substitution technique. In this technique RGB color channel component of cover image is used for LSB substitution. The cover is divided into several parts and a different scheme is used on each part of the cover image. The method uses three mechanisms corresponding to the three sequences. First is 'start from the corner' second is 'start from centre' and third is 'hybrid'. Two sequencing mechanisms used are counter clockwise direction and clock wise direction. This method only increases the security and embedding efficiency is not achieved.

Mishra et al. [4] suggested a new version involving LSB method. The secret binary message is first divided into 8-bit blocks and image cover is divided into 8-pixel blocks. A random number generator is used to select an 8-pixel block and the 8-bit message is embedded in it. The message is distributed to various cover image blocks. This method has a disadvantage of generating large meta-data. Swain and Lenka [5] used another technique where substitution takes place alternatively at 7^{th} and 8^{th} bit positions. First the information is encrypted and then embedded to the cover image. The first cover byte uses 7^{th} bit position to hide a message bit and second cover byte uses 8^{th} bit position, third cover byte again uses 7^{th} bit position and then 8^{th} bit position and so on. This scheme provides two levels of security but has lower embedding efficiency. Rosziati and Teoh Suk [6] use steganography imaging system (SIS) technique. The method is used to hide data in cover image. At the receiver end, reverse method is used to retrieve the secret data back using a key. During the process of hiding message data, a secret key is calculated. This method provides accuracy, privacy and confidentiality by the use of secret key but lacks embedding efficiency.

A moderate bit substitution method is proposed by Pharwaha [7], in which 4^{th} LSB cover bit is replaced by the secret data bits. After embedding data bit, a post pixel adjustment process is used to improve the quality of stego-image. This scheme only enhances the security and not embedding capacity. Jain and Ahirwal [8] use a private stego-key for embedding mechanism. The stego-key uses five dark level ranges. The chosen key uses five territories and substitutes diverse number of bits in LSBs. The technique covers up to four bits in a few pixels. The disadvantage is that the 4^{th} bit introduces greater distortion. R.Das and T. Tuithung [9] proposed an image steganography using Huffman encoding. Two pictures of size MxN and PxQ are used as cover pictures and a secret picture. Huffman encoding is performed over the secret message before it is embedded inside the cover picture. The disadvantage is that Huffman table and length of the bit stream of Huffman encoding is also inserted in the cover image. Ahn et al. [10] construct mathematical frameworks based on complex theoretic view to find the extents of steganography. This method lacks embedding efficiency. Pang [11] uses another construction; his scheme uses hash value that is obtained from a filename, password and locates the hidden header position but lacks embedding capacity. M. Dobsicek [12], develops another steganographic application where one key is used to encrypt the content whereas several keys can be used to decrypt the same content. The amount of information corresponds to the relative entropy between encrypt and decrypt key. This method lacks embedding efficiency. Machine learning techniques have been suggested to characterize images as non-suspicious or suspicious by Mittal et al. [13]. This method lacks efficient techniques to characterize images. Another entropy based technique detects the suitable areas in the image where information is embedded with minimum distortion Pavan et al. [14]. The method lacks embedding capacity. C. Zhang et al. [15] uses a technique to hide secured binary bits indirectly within some selected image bits. A neural network algorithm is used to get cipher bits. This method generates large metadata and hence lacks embedding capacity. The techniques discussed in [16], [19], [22], [23] uses LSB bit to hide the message. These techniques are limited to LSB bit and have low embedding capacity. The techniques mentioned by [17], [20], [21] have lower embedding efficiency as compared to proposed method. The technique presented by K. Hossain et al. [18] provides security of video files only. The PVD techniques discussed by [24], [25] does not provide good embedding efficiency in smooth area of image and has a maximum embedding capacity limit. The proposed algorithm is independent of image smoothness and embedding capacity can be increased to higher level by using bits above the LSB bit.

2. LSB embedding method

A bitmap file embedding uses a simple method to embed data into a cover file. In this method the file is extracted in a memory buffer where file header is separated from the rest of the pixel bytes. The secret data is then converted into stream of bits and it is embedded into the cover LSB bits. To embed two characters "A" and "B" for example, first these are converted into bits, then the bits directly overwrites the LSB bits of the cover image. Use of simple embedding method is easy at the cost of lesser embedding efficiency. The embedding efficiency can be further increased if additional techniques are adopted as discussed in the next section.

3. Proposed method

Normally it is difficult and inefficient to work with LSB's directly in the cover image. To make the process more efficient and practical LSB's of cover image are extracted into an array. The size of this array is 8-times lesser than the cover image. This extraction is one time process. Moreover the extracted array is 1/8 the size of full cover that results in lesser memory requirement and increases the speed of the algorithm. All the embedding space is represented as layers with minimum 1 to maximum 8 layers. Each layer is further divided into partitions from 1 to maximum 255.

The LSB of each byte of cover is copied in an array, represents the 1^{st} layer of bits or bytes. Similarly 2^{nd} LSB of each byte of cover is copied in an array represents 2^{nd} layer of bits or bytes. Similarly we generate 3^{rd} layer from 3^{rd} LSB of each byte of cover. The most significant bit of each byte generates the 8^{th} layer. The distortion is minimum when we hide data in the first layer. We use higher layers if data size exceeds the size of first layer. The value nLayers represents the total number of layers used to hide data. We divide each layer into equal parts with user defined value from 0 to 255. A value nparts[k] contains the total number of parts, k^{th} layer is divided. Another variable partsize[k] and datasize[k] contains size of partition and the size of data it contains in k^{th} layer. Offset[k][m] represents data offset or starting position of data in m^{th} partition of k^{th} layer. We take m as an integer value from 1 to 255. The maximum value of k and m is user defined and can be different for different layers.

4. Linear scan array matching

This technique gives results for better PSNR value as well as it provides security to the stego image. The data is converted into linear array of bits. The size of this array must be less than the size of the LSB's array extracted in the above process. The EXTRACT_LSB_BITS algorithm inputs full cover of an image as FULLCOVER, its size as NFULLCOVER and the total number of layers as NLAYERS to extract. The output bytes are represented by LSBCOVERBYTES with size equal to MAXBYTESPERLAYER. Algorithm-1 represents the complete working of the process.

Algorithm-1;

Extract_Lsb_Bits (Fullcover, Nfullcover, Nlayers, Lsbcoverbytes, Maxbytesperlayer)

1) Set Mask:=0
2) Set Byteno:=0
3) Repeat Step 4 For I=0,1,2…Nlayers*Maxbytesperlayer-1
4) Set Lsbcoverbytes[I]:=0
5) Repeat Steps 6 To 10 For I=0,1,2…Nlayers
6) Set Mask:=Power(2,I)
7) Repeat Steps 8 To 10 For J=0,1,2.. Maxbytesperlayer*8-1
8) Byteno=J/8+I*Maxbytesperlayer
9) Shift Bits Left Of Lsbcoverbytes[Byteno] By 1
10) If Fullcover[J] Bitwise And Mask Is Nonzero

Lsbcoverbytes[Byteno]= Lsbcoverbytes[Byteno] Bitwise Or 1

[End Of If Structure]
11) Return

The algorithm-2 EMBED_LSB_BITS embed the extracted cover LSB bits back to the cover image. This process is reverse of the extracting process. The parameters are same as that of extracting process.

Algorithm-2:

Embed_Lsb_Bits(Fullcover, Nfullcover, Nlayers, Lsbcoverbytes, Maxbytesperlayer)
1) Set Byteno:=0
2) Set Bitno:=0, Mask
3) Set Val[10]:= { 1,2,3,4,8,16,32,64,128,0 }
4) Set Val2[10]:={ 254,253,251,247,239,223,191,127}
5) Repeat Steps 7 To 12 For I=0,1,2…Nlayers-1
6) Set Mask=Power(2,I)
7) Repeat Steps 9 To 11 For J=0,1,2…Maxbytesperlayer*8
8) Set Byteno:=J/8+I*Maxbytesperlayer
9) Set Bitno:=7-(J%8)
10) Set Fullcover[J]:= Fullcover[J] Bitwiseand Val2[I]
11) If (Lsbcoverbytes[Byteno] Bitwise And Val2[I]) Is Nonzero
12) Fullcover[J]:=Val[I]
13) [End Of If Structure]
14) Return

The algorithm-3 rotates a data buffer to left of size TDATA-BYTES, starting at DATA position, through nROTATEBITS number of bits.

Algorithm-3:

ROTATE_BUFFER_LEFT(DATA, TDATABYTES, Nrotatebits)
1) Set Nrotatebytes:=Nrotatebits/8
2) Do While Nrotatebytes > TDATABYTES
Nrotatebytes=Nrotatebytes-TDATABYTES
[End Of While Structure]
3) If Nrotatebytes Is Not Equal To 0
Set K:=0 And I:=Nrotatebytes
Do While I<Nrotatebytes
TEMP[K]=DATA[I]
I=I+1
K=K+1
End Of While Structure]
I:=0
Do While I<TDATABYTES
DATA[I]=TEMP[I]
I=I+1
K=K+1
[End Of While Structure]
Set I:=0
Do While I<TDATABYTES
DATA[I]=TEMP[I]
I=I+1
[End Of Do While Structure]
[End Of If Structure]
4) Nrotatebits= Nrotatebits%8
5) If Nrotatebits=0 Then Return
6) Set MASK:=0 And I:=0
7) Do While I< Nrotatebits
MASK=MASK+Power(2,7-I)
I=I+1
[End Of Do While Structure]
8) Set I:=0
9) Do While I< TDATABYTES
TEMPVAR= DATA[I] Bitwise AND MASK
MASKARRAY[I]=TEMPVAR Bitshift Right By (8-Nrotatebits)
DATA[I]=DATA[I] Bitshift Left By Nrotatebits
I=I+1
[End Of Do While Structure]
10) Set I:=0
11) Do while i<TDATABYTES-1
DATA[I]=DATA[I] Bitwise-OR MASKARRAY[I+1]
I=I+1
[End of do while structure]
12) DATA[TDATABYTES-1]=DATA[TDATABYTES-1] Bitwise-OR MASKARRAY[0]
13) return

The algorithm-4 rotates a data buffer to right of size TDATA-BYTES, starting from DATA, through nROTATEBITS number of bits.

Algorithm-4

ROTATE_BUFFER_RIGHT(DATA, TDATABYTES, Nrotatebits)
1) Set Nrotatebytes:=Nrotatebits/8
2) Do While Nrotatebytes > TDATABYTES
Nrotatebytes=Nrotatebytes-TDATABYTES
[End Of Do While Structure]
3) If Nrotatebytes Is Not Equal To 0
Set K:=0 And I:=Nrotatebytes
Do While I<Nrotatebytes
TEMP[I]=DATA[K]
I=I+1
K=K+1
[End Of While Structure]
I:=0
Do While K<TDATABYTES
TEMP[I]=DATA[K]
I=I+1
K=K+1
[End Of While Structure]
Set I:=0
Do While I<TDATABYTES
DATA[I]=TEMP[I]
I=I+1
[End Of Do While Structure]
[End Of If Structure]
4) Nrotatebits= Nrotatebits%8
5) If Nrotatebits=0 Then Return
6) Set MASK:=0 And I:=0
7) Do While I< Nrotatebits
MASK=MASK+Power(2,I)
I=I+1
[End Of Do While Structure]
8) Set I:=0
9) Do While I< TDATABYTES
TEMPVAR= DATA[I] Bitwise-AND MASK
MASKARRAY[I]=TEMPVAR Bit-Shift-Left By (8-Nrotatebits)
DATA[I]=DATA[I] Bit-Shift-Right By Nrotatebits
I=I+1
[End Of Do While Structure]
10) Set I:=0
11) Do While I<TDATABYTES-1
DATA[I]=DATA[I] Bitwise-OR MASKARRAY[I+1]
I=I+1
[End Of Do While Structure]
12) DATA[TDATABYTES-1]=DATA[TDATABYTES-1] Bitwise-OR MASKARRAY[0]
13) Return

The first layer of bits extracted from cover can be represented in the form of partitions as shown in figure 1. The partitions are equal in size and the total number of partitions is optimized to get the maximum embedding efficiency.
The secret data is divided into equal parts. Each data part is rotated over the corresponding partition to find the best matching posi-

tion within each partition. We find the best data starting position where it matches to the maximum extent. The starting positions are saved in offset locations. Different layers can have different number of partitions. Figure 2 shows an example of two partitions in the second layer. The meta-data is stored at the end of the layer. Depending upon the total number of partitions, size of meta-data is generated. If the size of data is lesser than the total size of partitions then the data is equally divided to the total number of partitions. Figure 3 shows a single partition is created in a single layer of bits. A single partition a good option when the size of secret data is small that generates smaller meta-data. Figure 4 shows a general view of partitions and the different meta-data variables required to recover the embedded data from cover image. The meta data is stored at the end of last partition as shown in figure 4.

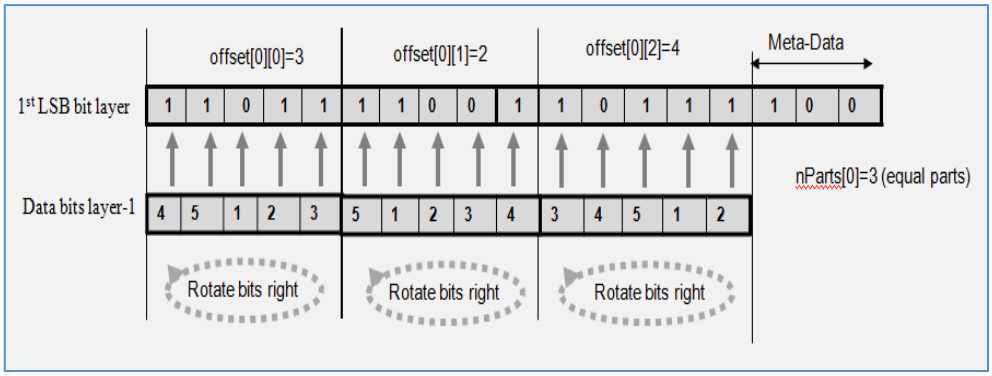

Fig. 1: Three Equal Partitions of First LSB Bit Layer. Offset Stores the Starting Position of Data in the Partition.

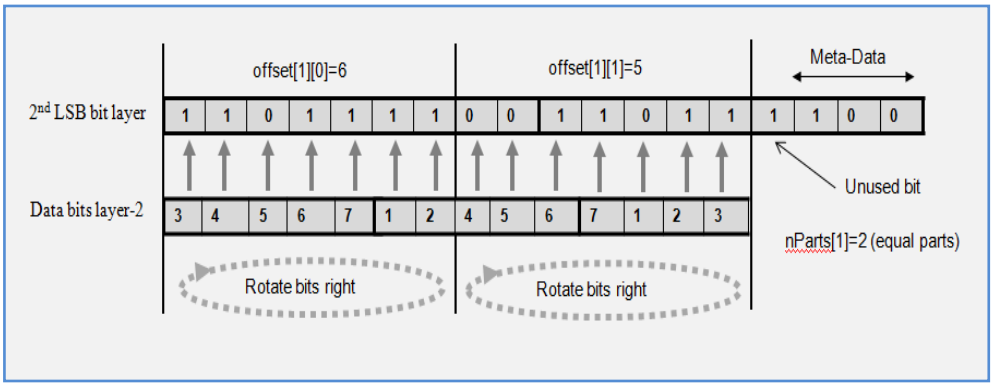

Fig. 2: Two Equal Partitions of Second LSB Bit Layer.

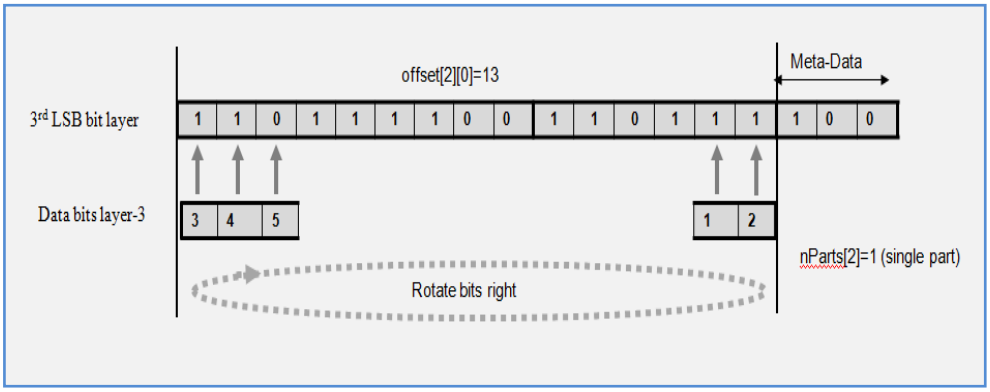

Fig. 3: A Single Partition in Third LSB Bit Layer.

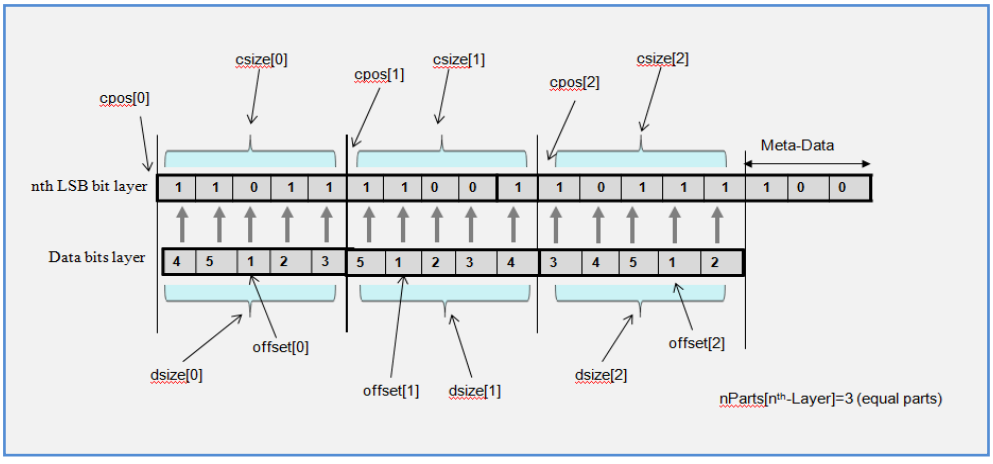

Fig. 4: A General Structure of Partitions within A Cover Bits Layer and Data Bits Along with Different Types of Variables That Represent the Meta-Data Information.

Table 1: Comparison of Results at Zero Offset and at Best Offset Using 50 Partitions

Part No.	Data Size (bits)	Cover Size (bits)	Maximum Data bit Matches	Minimum MisMatches	Best Data Offset Position	Percentage matches at Zero Offset	Percentage Maximum Matches (At Best offset)
1	656	15712	378	278	7334	49.39%	57.62%
2	656	15712	383	273	2999	50.61%	58.38%
3	656	15712	381	275	3677	50.15%	58.08%
4	656	15712	376	280	5070	51.68%	57.32%
5	656	15712	376	280	2111	53.96%	57.32%
6	656	15712	377	279	9322	50.91%	57.47%
7	656	15712	382	274	11670	50.76%	58.23%
8	656	15712	381	275	15320	51.52%	58.08%
9	656	15712	374	282	14684	47.87%	57.01%
10	656	15712	377	279	5680	51.07%	57.47%
11	656	15712	379	277	930	47.71%	57.77%
12	656	15712	381	275	8329	53.35%	58.08%
13	656	15712	378	278	7129	48.17%	57.62%
14	656	15712	380	276	6580	52.74%	57.93%
15	656	15712	382	274	15078	49.54%	58.23%
16	656	15712	382	274	8176	51.37%	58.23%
17	656	15712	377	279	12626	47.41%	57.47%
18	656	15712	381	275	5205	50.61%	58.08%
19	656	15712	375	281	3955	51.07%	57.16%
20	656	15712	373	283	710	50.91%	56.86%
21	656	15712	378	278	9035	49.24%	57.62%
22	656	15712	376	280	15236	50.00%	57.32%
23	656	15712	388	268	14481	46.65%	59.15%
24	656	15712	379	277	4191	48.78%	57.77%
25	656	15712	384	272	12816	48.02%	58.54%
26	656	15712	372	284	7218	48.17%	56.71%
27	656	15712	379	277	1859	52.44%	57.77%
28	656	15712	379	277	11789	48.93%	57.77%
29	656	15712	380	276	15610	48.63%	57.93%
30	656	15712	378	278	9772	48.02%	57.62%
31	656	15712	378	278	3024	51.07%	57.62%
32	656	15712	379	277	2244	50.15%	57.77%
33	656	15712	374	282	10879	51.07%	57.01%
34	656	15712	376	280	13666	52.29%	57.32%
35	656	15712	383	273	5405	47.56%	58.38%
36	656	15712	386	270	4630	53.35%	58.84%
37	656	15712	374	282	7492	52.29%	57.01%
38	656	15712	387	269	5964	49.85%	58.99%
39	656	15712	372	284	2053	50.61%	56.71%
40	656	15712	375	281	6334	51.52%	57.16%
41	656	15712	379	277	12973	50.30%	57.77%
42	656	15712	381	275	8766	48.17%	58.08%
43	656	15712	373	283	10446	50.46%	56.86%
44	656	15712	384	272	4457	45.12%	58.54%
45	656	15712	374	282	8434	47.87%	57.01%
46	656	15712	377	279	9212	50.30%	57.47%
47	656	15712	376	280	9558	46.04%	57.32%
48	656	15712	378	278	3452	50.61%	57.62%
49	656	15712	376	280	3954	50.76%	57.32%
50	624	15712	356	268	6294	47.44%	57.05%

5. Results

Table 1 shows the LSB layer is divided into 50 partitions. We have taken a random secret data of 4KB. A 768kB size of 24-bit .BMP cover image is used. The LSB layer of bits is extracted from cover image and is converted into partitions. Each partition is 15712 bits and data of 656 bits is inserted at best data offset position. Percentage matching of data embedded at zero offset is compared with that embedded at best offset. The results indicate a better matching and hence lower distortion of cover image that further results to lower MSE value and greater PSNR value. Similarly Table 2 and Table 3 shows with partitions of size 25 and 10 respectively. As the number of partitions decreases, the percentage matching also decreases. As the number of partitions increases the data is equally divided into smaller parts that increase the percentage of matching. Increasing the number of partitions increases the size of meta-data that needs to be stored at the end of partitions for recovery of secret data. An optimal value of number of partitions is obtained from two input values. First is the size of cover image layer and second is total secret data to embed in the layer. Figure 5 shows comparison between two embedding schemes. The first is normal embedding that do not use rotate and match algorithm and uses only partitions that break cover image layer and secret data into different parts. The second method uses rotate and match algorithm in addition to partitions on a layer. The LSB layer is divided into 50 partitions (samples). The percent match of data within each partition for both methods is shown in figure 5. The plot clearly indicates that for each partition, there is an increase of percentage match of data bits from 8% to 10%.

Table 2: Comparison of Results at Zero Offset with the Best Offset Using 25 Partitions

Part No.	Data Size (bits)	Cover Size (bits)	Maximum Data bit Matches	Minimum MisMatches	Best Data Offset Position	Percentage matches at Zero Offset	Percentage Maximum Matches (at Best offset)
1	1312	31440	733	579	25406	49.16%	55.87%
2	1312	31440	731	581	8294	50.08%	55.72%
3	1312	31440	724	588	21409	49.92%	55.18%
4	1312	31440	728	584	11414	45.35%	55.49%
5	1312	31440	728	584	12969	49.85%	55.49%
6	1312	31440	739	573	23305	48.86%	56.33%
7	1312	31440	737	575	21748	48.63%	56.17%
8	1312	31440	725	587	25908	50.91%	55.26%
9	1312	31440	726	586	6775	47.79%	55.34%
10	1312	31440	725	587	3187	49.92%	55.26%
11	1312	31440	726	586	13806	49.39%	55.34%
12	1312	31440	737	575	14305	48.48%	56.17%
13	1312	31440	741	571	12624	50.46%	56.48%
14	1312	31440	736	576	1203	51.14%	56.10%
15	1312	31440	724	588	21454	48.25%	55.18%
16	1312	31440	719	593	28801	50.08%	54.80%
17	1312	31440	725	587	6623	50.38%	55.26%
18	1312	31440	730	582	25223	50.53%	55.64%
19	1312	31440	732	580	21356	47.48%	55.79%
20	1312	31440	725	587	4012	49.39%	55.26%
21	1312	31440	727	585	6306	49.62%	55.41%
22	1312	31440	731	581	9694	53.35%	55.72%
23	1312	31440	720	592	8731	48.17%	54.88%
24	1312	31440	726	586	21081	50.30%	55.34%
25	1280	31440	708	572	14635	49.45%	55.31%

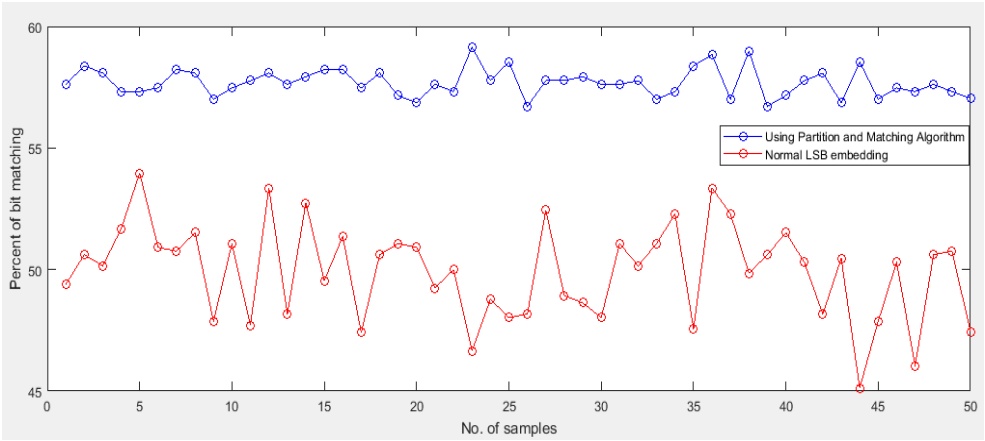

Fig. 5: Comparison of Normal Embedding Scheme with Scheme Using Partition and Matching Algorithm.

Table 3: Comparison of Results at Zero Offset with the Best Offset Using 10 Partitions

Part No.	Data Size (bits)	Cover Size (bits)	Maximum Data bit Matches	Minimum MisMatches	Best Data Offset Position	Percentage matches at Zero Offset	Percentage Maximum Matches (at Best offset)
1	3280	78624	1763	1517	24070	49.12%	53.75%
2	3280	78624	1759	1521	18185	50.46%	53.63%
3	3280	78624	1759	1521	75003	50.85%	53.63%
4	3280	78624	1773	1507	51363	50.21%	54.05%
5	3280	78624	1756	1524	24918	51.86%	53.54%
6	3280	78624	1743	1537	16771	50.40%	53.14%
7	3280	78624	1752	1528	65808	50.06%	53.41%
8	3280	78624	1760	1520	36876	48.02%	53.66%
9	3280	78624	1759	1521	11301	51.62%	53.63%
10	3248	78624	1742	1506	13	50.92%	53.63%

6. Conclusion

This paper presents a novel method for data hiding based on LSB and higher bits of cover bytes. From the results obtained in figure 5, it is concluded that the total number of matches increases as the total number of partitions increases. The data is split into smaller chunks there by increasing the match percentage. This reduces MSE of cover image and improves the PSNR value of the stego images. The embedding capacity is increased by using higher bit layers above the LSB bit layer. The method first starts hiding data in low distortion areas of image and hides further to increase the embedding capacity with higher embedding efficiency. The security of hidden message is automatically inherited. The meta-data generated is directly proportional to the number of partitions. The optimal value of number of partitions depends upon the size of LSB layer and the total size of secret data to embed in single layer. The proposed method gives 8% to 10% better results that vary from partition to partition as compared to existing methods.

References

[1] Ki-Hyun Jung, Kee-Young Yoo, “Data hiding using run length matching”, International Journal of Intelligent Information and Database Systems 2009 - Vol. 3, No.3 pp. 311 - 325. https://doi.org/10.1504/IJIIDS.2009.027689.

[2] Kekre, Hemant & Athawale, Archana & Halarnkar, Pallavi. (2008). Increased Capacity of Information Hiding in LSB's Method for Text and Image. International Journal of Electrical, Computer, and Systems Engineering.

[3] H. Mathkour, G. M. R. Assassa, A. A. Muharib and I. Kiady, "A Novel Approach for Hiding Messages in Images," 2009 International Conference on Signal Acquisition and Processing, Kuala Lumpur, 2009, pp. 89-93. https://doi.org/10.1109/ICSAP.2009.36.

[4] Mishra, A. Gupta and D. K. Vishwakarma, "Proposal of a New Steganographic Approach," 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, Trivandrum, Kerala, 2009, pp.175-178. https://doi.org/10.1109/ACT.2009.52.

[5] G.Swain and S.K. Lenka, “steganography-Using a Double Substitution Cipher.” International Journal of Wireless Communications and Networking, Vol. 2, No. 1, pp.35-39, 2010.

[6] Ibrahim, Rosziati & Suk Kuan, Teoh. (2010). Steganography Imaging System (SIS): Hiding Secret Message inside an Image. Lecture Notes in Engineering and Computer Science. 2186.

[7] A.P.S Pharwaha, “Secure Data Communication using Moderate Bit Substitution for Data Hiding with Three Layer Security” IE (I) Journal-ET, Vol. 91, pp.45-50, 2010.

[8] Y.K. Jain and R.R. Ahirwal, “A Novel Image Steganography Method With Adaptive Number of Least Significant Bits Modification Based on Private Stego-Keys” International Journal of Computer Science and Security, Vol.4, No. 1, pp.40-49, 2010.

[9] R.Das and T. Tuithung, “A Novel Steganography Method for Image Based on Huffman Encoding” IEEE, ISBN: 978-1-4577-0748-3, 2012.Dumitrescu, S., W. Xiaolin and Z. Wang, "Detection of LSB steganography via sample pair analysis," In: LNCS, Springer-Verlag, New York, Vol. 2578/2003, pp: 355–372, 2003.

[10] Ahn, L.V. and N.J. Hopper, "Public-key steganography. In Lecture Notes in Computer Science of Advances in Cryptology," EUROCRYPT 2004, Vol. 3027 / 2004, Springer-Verlag Heidelberg, pp: 323–341, 2004. https://doi.org/10.1007/978-3-540-24676-3_20.

[11] Pang, H.H., K.L. Tan and X. Zhou, "Steganographic schemes for file system and b-tree," IEEE Trans. on Knowledge and Data Engineering, Vol. 16, pp.701–713, 2004. https://doi.org/10.1109/TKDE.2004.15.

[12] Dobsicek, M., "Extended steganographic system," In: 8th Intl. Student Conf. on Electrical Engineering. FEE CTU. 2004

[13] Mittal, U. and N. Phamdo, "Hybrid digital-analog joint source-channel codes for broadcasting and robust communications," IEEE Trans. On Info. Theory, vol. 48, pp. 1082 –1102, 2002. https://doi.org/10.1109/18.995544.

[14] Pavan, S., S. Gangadharpalli and V. Sridhar, "Multivariate entropy detector-based hybrid image registration algorithm," IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, pp: 18-23, 2005.

[15] C. Zhang, H.W. Guesgen, W.K. Yeap "Neural Based Steganography, Lecture note in computer science Computational Intelligence. Neural Networks," LNAI 3157, pp. 429–435, Springer-Verlag Berlin Heidelberg 2004. https://doi.org/10.1007/978-3-540-28633-2_46.

[16] K. A. Al-Afandy, O. S. Faragallah, A. Elmhalawy, E. S.M. El-Rabaie and G. M. El-Banby, "High security data hiding using image cropping and LSB least significant bit steganography," 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), Tangier, 2016, pp. 400-404. https://doi.org/10.1109/CIST.2016.7805079.

[17] Da-Chun Wu and Wen-Hsiang Tsai. 2003. A steganographic method for images by pixel-value differencing. *Pattern Recogn. Lett.* 24, 9-10 (June 2003), 1613-1626. https://doi.org/10.1016/S0167-8655(02)00402-6.

[18] K. Hossain and R. Parekh, "An approach towards image, audio and video steganography," *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Kolkata, 2016, pp. 302-307. https://doi.org/10.1109/ICRCICN.2016.7813675.

[19] C. C. Chang and H. W. Tseng, "Data Hiding in Images by Hybrid LSB Substitution," *2009 Third International Conference on Multimedia and Ubiquitous Engineering*, Qingdao, 2009, pp. 360-363. https://doi.org/10.1109/MUE.2009.68.

[20] K. Joshi and R. Yadav, "New approach toward data hiding using XOR for image steganography," *2016 Ninth International Conference on Contemporary Computing (IC3)*, Noida, 2016, pp. 1-6. https://doi.org/10.1109/IC3.2016.7880204.

[21] Gambhir and S. Khara, "Integrating RSA cryptography & audio steganography," *2016 International Conference on Computing, Communication and Automation (ICCCA)*, Noida, 2016, pp. 481-484. https://doi.org/10.1109/CCAA.2016.7813767.

[22] M. Hussain and M. Hussain, "Pixel intensity based high capacity data embedding method," *2010 International Conference on Information and Emerging Technologies*, Karachi, 2010, pp. 1-5. https://doi.org/10.1109/ICIET.2010.5625723.

[23] M. Sabokdast and M. Mohammadi, "A steganographic method for images with modulus function and modified LSB replacement based on PVD," *The 5th Conference on Information and Knowledge Technology*, Shiraz, 2013, pp. 121-126. https://doi.org/10.1109/IKT.2013.6620050.

[24] F. Pan, J. Li and X. Yang, "Image steganography method based on PVD and modulus function," *2011 International Conference on Electronics, Communications and Control (ICECC)*, Zhejiang, 2011, pp. 282-284. https://doi.o.rg/10.1109/ICECC.2011.6067590.

[25] Z.Hanling, G. and X. Caiqiong, "Image Steganography Using Pixel-Value Differencing," ," *2009 Second International Symposium on Electronic Commerce and Security*, Nanchang, 2009, pp. 109-112. https://doi.org/10.1109/ISECS.2009.139.