International Journal of Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-5, Issue-7)

Research Article | **July 2016**

Performance Analysis of Data Compression Techniques for Multimedia Data Hiding

M. Senthilkumar*
Research Scholar, AMET University,
Chennai, Tamilnadu, India

V. Mathivanan
Research Supervisor, AMET University,
Chennai, Tamilnadu, India

Abstract—

The Multimedia Data Compression Techniques regarding to embed in Information hiding aspect are implemented. For solving these data compression techniques and to test the computation efficiency (compression ratio, compression time) of the algorithm computer coded have been developed. Data compression is often referred to as coding, where coding is a very general term encompassing any special representation of data which satisfies a given need. Information theory is defined to be the study of efficient coding and its consequences, in the form of speed of transmission and probability of error. I consider sample data and comparing with the mentioned four different data compression techniques.

Keywords— Compression, Information hiding, Transmission, Probability and Encompassing

I. INTRODUCTION

Data compression has important application in the areas of data transmission and data storage. Many data processing applications require storage of large volumes of data, and the number of such applications is constantly increasing as the use of computers extends to new disciplines. At the same time, the proliferation of computer communication networks is resulting in massive transfer of data over communication links. Compressing data to be stored or transmitted reduces storage and/or communication costs. When the amount of data to be transmitted is reduced, the effect is that of increasing the capacity of the communication channel. Similarly, compressing a file to half of its original size is equivalent to doubling the capacity of the storage medium. It may then become feasible to store the data at a higher, thus faster, level of the storage hierarchy and reduce the load on the input/output channels of the computer system.

II. LITERATURE SURVEY

An In general, data compression consists of talking a stream of symbols and transforming them into codes. If the compression is effective, the resulting stream of codes will be smaller than the original symbols. The decision is output a certain code for a symbol or set of symbols is based on a model. So, data compression consists of two components, modeling and coding.

Crypto graphic enabled Compression of Multimedia content [7][8] is the art of hiding and transmitting data through apparently innocuous carriers in an effort to conceal the existence of the data, the word Steganography literally means covered or hiding writing as derived from Greek. Steganography has its place in security. It is not intended to replace cryptography but supplement it. Hiding a message with Steganography methods reduces the chance of a message being detected. If the message is also encrypted then it provides another layer of protection.

Therefore, some Steganographic methods combine traditional Cryptography with Steganography; the sender encrypts the secret message prior to the overall communication process, as it is more difficult for an attacker to detect embedded cipher text in a cover. Hidden information in the cover data is known as the "embedded" data and information hiding is a general term encompassing many sub disciplines [9][10], is a term around a wide range of problems beyond that of embedding message in content.

The term hiding here can refer to either making the information undetectable or keeping the existence of the information secret. Information hiding is a technique of hiding secret using redundant cover data such as images, audios, movies, documents, etc.

This technique has recently become important in a number of application areas. For example, digital video, audio, and images are increasingly embedded with imperceptible marks, which may contain hidden signatures or watermarks that help to prevent unauthorized copy[11][12].

Many different methods enable hiding information in audio and image. These methods may include hiding information in unused space in file headers to hold ‘extra’ information. Embedding techniques can range from the placement of information in imperceptible level[13][14] (noise), manipulation of compression algorithms, and the modification of carrier properties. In audio, small echoes or slight delays can be added or subtle signals can be masked by sound of higher amplitude. Information (data) can be hidden in different ways in image. To hide information; straight message insertion may encode every bit of information in the image or selectively embed the message in busy areas where it would be less perceptible. A message may also be scattered randomly or repeated several times throughout the image since the advent of computers there has been a vast dissemination of information, some of which needs to be kept private, some of which doesn't.

The information may be hidden in two basic ways (Cryptography and Steganography).The methods of Cryptography does not conceal the presence of secret information but render it unintelligible to outsider by various transformations of the information that is to be put into secret form, while methods of Steganography conceal the very existence of the secret information [16][17]. To encrypt messages longer than the block size (128 bits in the above example), a mode of operation is used [18][19]. Block ciphers can be contrasted with stream ciphers; a stream cipher operates on individual digits one at a time and the transformation varies during the encryption.

2.1 Multimedia Data Compression in Data mining perspective

Data mining is an evolving field with new concepts born monthly and current concepts struggling to retain their place. Researchers from different branches of mathematics, statistics, marketing, or artificial intelligence will use different terminologies[3][31][1] Where a statistician sees dependent and independent variables, and an artificial intelligence researcher sees features and attributes, others see records and fields (Berryand Linoff, 1997).

The phrase "neural networks" is synonymous with data mining. Although data mining is known for having exotic names, the field has begun to include certain kinds of descriptive statistics and visualization techniques into data mining (Westphal and Blaxton, 1998). Stat soft, an on-line statistical software provider, seemed to favour "exploratory data analysis." Berthold and Hand (1999) called their work "intelligent data analysis."

Data mining assumes the existence of spherical multi-Euclidean dimensions (Delmater and Hancock, 2001). The *n*-dimensional Euclidean space[32][33], or Euclidean hyperspace, is called feature space, where any given coordinates of ordered triples or ordered pairs are viewed as feature vectors. The understanding of Gaussian distribution, *z*-scores, and regression equations is very useful in data mining. One of the fundamental concepts operating within the data mining hyperspace is the cluster, which is formed of sets of feature vectors that are understood by examining their standard deviations. The tighter the vectors cluster, the better it is for classification purposes. In this case, the clusters are considered as good features, or gestalts.

Basic substitution systems try to encode secret information by substituting insignificant parts of the cover by secret message bits. The receiver can extract the information if he has knowledge of the positions where secret information has been embedded. Since only minor modifications are made in the embedding process, the sender assumes that they will not be noticed by an attacker [34][35].

It consists of several techniques that will be discussed in more detail, in the following subsection. Image downgrading is a special case of a substitution system in which image acts both as a secret message and a cover. Given cover-image and secret image of equal dimensions, the sender exchanges the four least significant bits of the cover gray scale (or color) values with the four most significant bits of the secret image[28][29].

III. DATA COMPRESSION TECHNIQUES

3.1 Huffman Compressor

The Huffman compressor can be implemented using two programs, one for compression and another for decompression. Also known as Huffman encoding, an algorithm for the lossless compression of files based on the frequency of occurrence of a symbol in the file that is being compressed. The Huffman algorithm is based on statistical coding, which means that the probability of a symbol has a direct bearing on the length of its representation. The more probable the occurrence of a symbol is, the shorter will be its bit-size representation. In any file, certain characters are used more than others. Using binary representation, the number of bits required to represent each character depends upon the number of characters that have to be represented. Using one bit we can represent two characters, i.e., 0 represents the first character and 1 represents the second character. Using two bits we can represent four characters, and so on.

Unlike ASCII code, which is a fixed-length code using seven bits per character, Huffman compression is a variable-length coding system that assigns smaller codes for more frequently used characters and larger codes for less frequently used characters in order to reduce the size of files being compressed and transferred.

For example, in a file with the following data:

XXXXXXYYYYZZ

The frequency of "X" is 6, the frequency of "Y" is 4, and the frequency of "Z" is 2. If each character is represented using a fixed-length code of two bits, then the number of bits required to store this file would be 24, i.e., (2 x 6) + (2x 4) + (2x 2) = 24.

If the above data were compressed using Huffman compression, the more frequently occurring numbers would be represented by smaller bits, such as:

X by the code 0 (1 bit)
Y by the code 10 (2 bits)
Z by the code 11 (2 bits)

Therefore the size of the file becomes 18, i.e., (1x 6) + (2 x 4) + (2 x 2) = 18.

In the above example, more frequently occurring characters are assigned smaller codes, resulting in a smaller number of bits in the final compressed file.

Huffman compression was named after its discoverer, David Huffman.

Huffman Decompression Program: This is Huffman Decompression Program. This program decompresses a file previously compressed with the HUFFCOMP.C program. Here the input direct from disk and output direct to disk (DISK to DISK).

3.2 Arithmetic Compressor

The arithmetic coding compressor can be implemented using two programs, one for compression and another for decompression.

Arithmetic Compression Program: This module is the program for a variable order compression program. This program also monitors compression ratios, and compression time. This program has to initialize the coder, the bit oriented I/O, the standard I/O. It then sits in a loop reading input symbols and encoding them. One difference is that every 256 symbols a compression check is performed. If the compression ratio exceeds 90%, a flush character is encoded. This flushes the encoding model, and will cause the decoder to flush its model when the file is being expanded.

Arithmetic Decompression Program: This module is the for a variable order finite context expansion program. The maximum order is determined by command line option. This program can respond to the FLUSH code inserted in the bit stream by the compressor. This routine first has to initialize the standard i/o, the bit oriented i/o, the arithmetic coder. The decompression loop differs in a couple of respects. First of all, it handles the special ESCAPE character, by removing them from the input bit stream but just throwing them away otherwise.

3.3 LZ78 Compressor

The way the dictionary is built during the encoding is expressed in the following Algorithm.

```
1. B ←Empty buffer of symbol
2: D ←Empty dictionary
3: while not End − Of − F ile do
4: S ←the next symbol from the input stream
5: I ← index of B in D
6: B.append(S)
7: if B is not in D then
8: Add B to the end of the dictionary
9: Output the two symbols < I, S >
10: B ← Empty buffer
11: end if
12: end while
13: if not B is empty then
14: I ← index of B without its last symbol in D
15: Output the two symbols < I, S >
16: end if
```

3.4 LZW Compressor

This is the LZW data compression/expansion program. This program gets a file name from the command line. It compresses the file, placing its output in a file named test.lzw. It then expands test.lzw into test.out. Test.out should then be an exact duplicate of the input file.

IV. ANALYSIS OF COMPRESSION TECHNIQUES

In this topic, the performance of each algorithm could be improved significantly beyond the implementations discussed here.

4.1 Huffman Compressor

Huffman coding, compression and decompression is a short program that needs very little memory. But it doesn't compress particularly well when compared to commonly use other compression techniques programs. The efficiency of Huffman coding also depends heavily on having a good estimate of the true probability of the value of each input symbol.

Improvements: Here to surpass the compression ratio of these programs, we need to start adding enhancements to the modeling code. That is Arithmetic coding produces slight gains over Huffman coding. So after constructing the Huffman code, once again apply the arithmetic coding in huffman code, might be we can get the better result.

4.2 Arithmetic Compressor

This algorithm wills results in both faster updates as well as better compression. This scheme works well for incrementally encoding a message.

There is enough accuracy retained during the double precision integer calculations to ensure that the message is accurately encoded. In order to use arithmetic coding to compress data, a model for the data is needed. The model needs to be able to do two things to effectively compress data:

- The model needs to accurately predict the frequency/probability of symbols in the input data stream.
- The symbol probabilities generated by the model need to deviate from a uniform distribution.

The process of encoding and decoding a stream of symbols using arithmetic coding is not too complicated. But at first glance, it seems completely impractical. Most computers support floating point numbers of up to 80 bits or so.

4.3 LZ78 Compressor

The LZ78 algorithm maintains a separate dictionary. Suppose we want to compress the following string of text:

the quick brown fox jumps over the lazy dog.

The word 'the' occurs twice in the file so this string is put in an index that is added to the compressed file and this entry is referred to as *****. The data then look like this: ***** *quick brown fox jumps over* ***** *lazy dog.*
This algorithm could be applied today to circumstances where either storage or transmission costs are extremely high.

4.4 LZW Compressor

LZW compression excels when confronted with data streams that have any type of repeated strings. Because of this, it does extremely well when compressing English text. Compression levels of 50% or better should be expected. Likewise, compressing saved screens and displays will generally show very good results.

Trying to compress binary data files is a little more risky. Depending on the data, compression may or may not yield good results. Highly redundant database files can be compressed down to 10% of their original size.

V. SAMPLE INPUT AND OUTPUT OF THE COMPRESSION TECHNIQUES

The outputs obtain in the coding are present in this section.

Huffman Coding: Run the program as: Huffcomp aaa.txt huff.in

```
Huffman Code Compression Program.
Getting Frequency Counts.
(1) Building Initial Heap.
(2) Building the Code Tree.
(3) Generating the Code Table.
(4) Compressing & Creating the Output File.

        Input characters (Bytes)          : 589312
        Compressed characters(Bytes)      : 340021
        Percentage Saving(ratio)          : 42.30%
        Compression time                  : 0.08 seconds
Huffman Code Decompression Program.
        Building the tree
        1. Decompressing & Creating the Output File
                Expansion time    : 0.09seconds
```

Arithmetic Coding: Run the program as: Arithcomp aaa.txt text.cmp

```
        Input characters (Bytes)          : 589312
        Compressed characters(Bytes)      : 327837
        Percentage Saving(ratio)          : 44.37%
        Compression time                  : 0.09 seconds
Decoding test.cmp to test.out
                Expansion time            :0.08 seconds
```

LZ78 coding: Run the program as LZ78 e aaa.txt lz78.in

```
        Input characters (Bytes)          : 589312
        Compressed characters(Bytes)      : 247837
        Percentage Saving(ratio)          : 57.94%
        Compression time                  : 0.10 seconds
Decoding as follows
        LZ78 d lz78.in lz78.out
                Expansion time            :0.06 seconds
```

LZW Coding: The sample input and output are obtained running this technique as

```
        Input file name?    aaa.txt
        Compressing ……
        Expanding ……
        Input characters (Bytes)          : 589312
        Compressed characters(Bytes)      : 280208
        Percentage Saving(ratio)          : 52.45%
        Compression time                  : 0.06 seconds
        Expansion time                    : 0.04 seconds
```

VI. TABLES AND CHARTS

The statistics for compression ratio, compression time and expansion time are presented in this section in the form of tables and charts.

Table 1 Compression Table

Compression Techniques	Size of Input file	Size of Compressed File	Compression Ratio	Compression Time (Comp. +Expan. Time)
Huffman Coding	589312 bytes	340021 bytes	42.30%	0.08 + 0.09 sec.
Arithmetic Coding	589312 bytes	327837 bytes	44.37%	0.09 + 0.08 sec.
LZ78 Coding	589312 bytes	247837 bytes	57.94%	0.10+0.06 sec
LZW Coding	589312 bytes	280208 bytes	52.45%	0.06+0.04 sec.

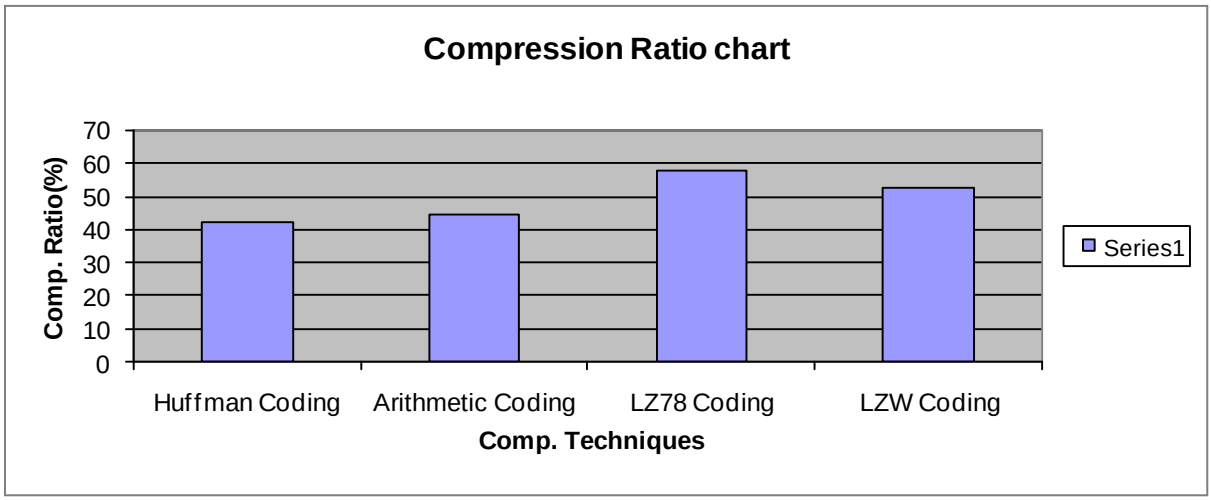

Fig. 1 Compression Ratio Chart

Table 2 Compression Time Chart

Compression Techniques	Compression Time
Huffman Coding	0.08 seconds
Arithmetic Coding	0.09 seconds
LZ78 Coding	0.10 seconds
LZW Coding	0.06 seconds

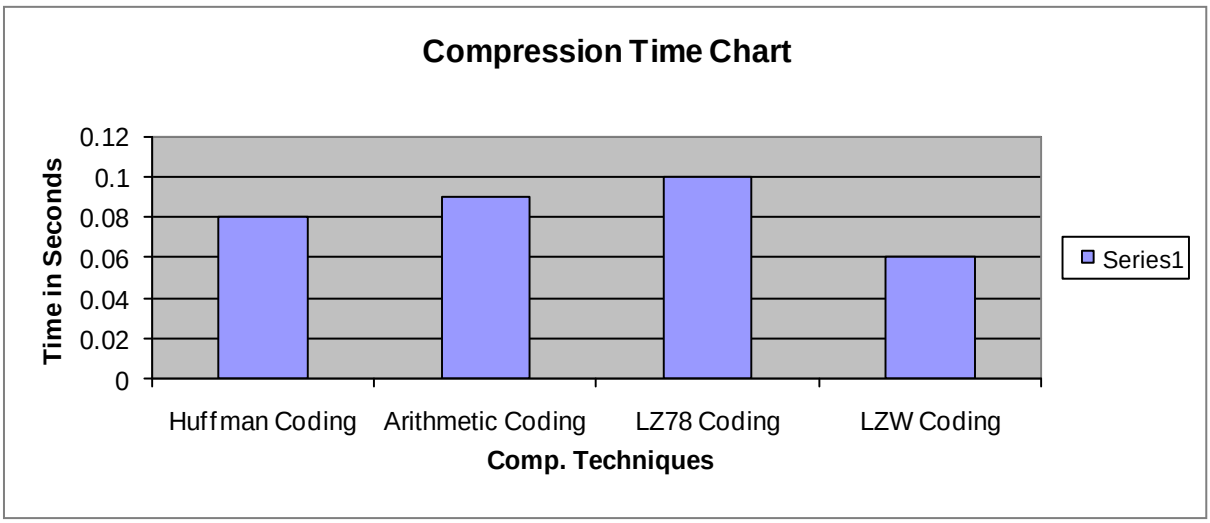

Fig. 2 Compression Time Chart

Table 3 Expansion time chart

Compression Techniques	Expansion Time
Huffman Coding	0.09 seconds
Arithmetic Coding	0.08 seconds
LZ78 Coding	0.06 seconds
LZW Coding	0.04 seconds

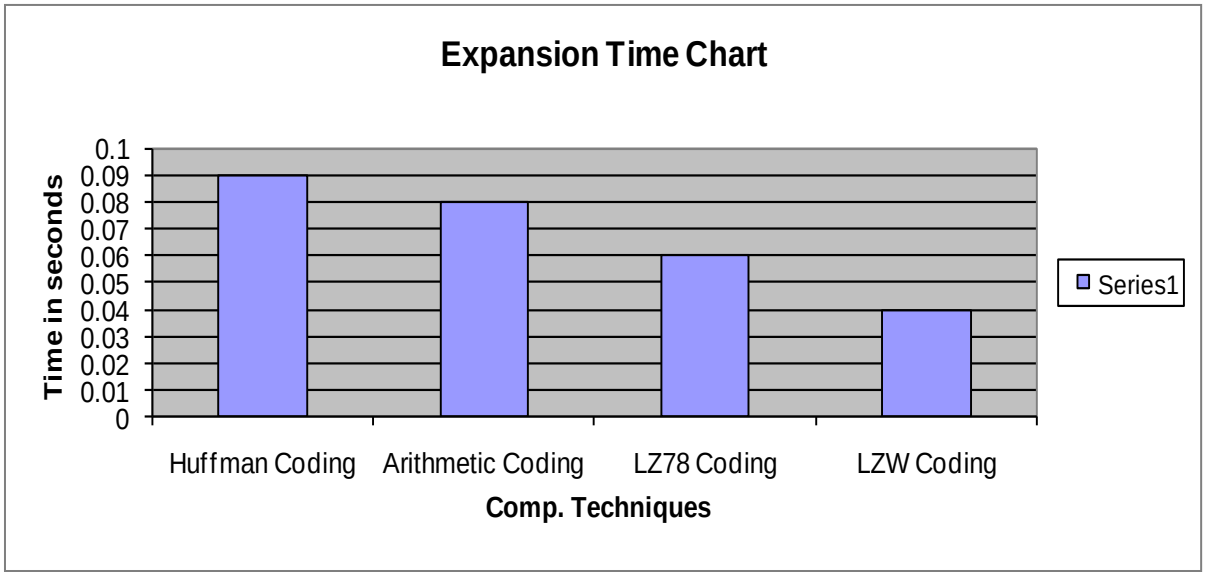

Fig. 3 Expansion Time Chart

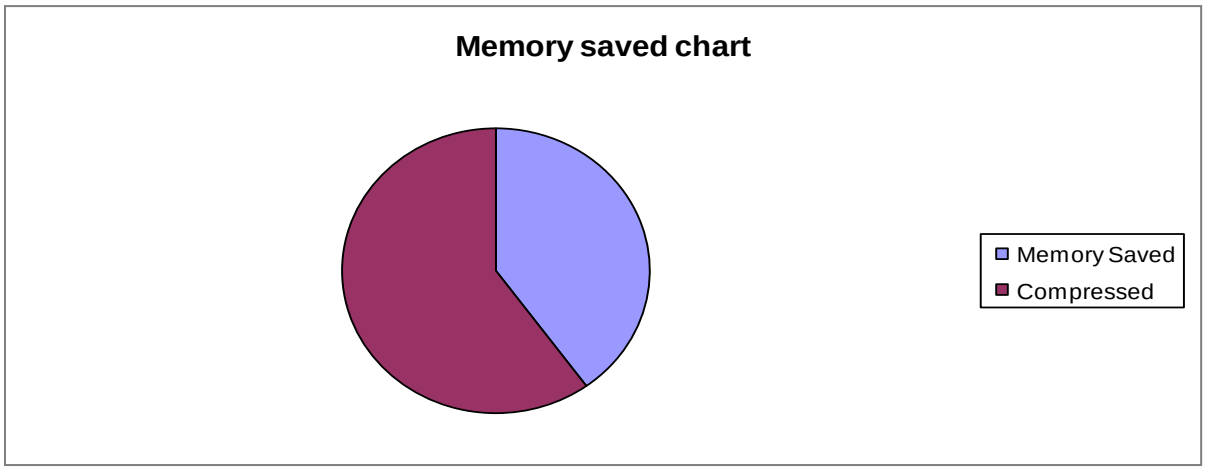

Fig. 4 Memory Saved Chart

That is,

Original	**Compressed**	**Memory saved**
100%	60%	40%

VII. CONCLUSIONS

Multimedia is a technology used to process textual data, audios and videos, images and animations. Multimedia applications are used in different fields of life like entertainment, video conferencing, application sharing, distance learning, remote working and online games. The different category of compression algorithm regarding to compress the multimedia content performance analysis is discussed in the above sections. Each and every section is illustrated the merits and demerits based on the performance evaluation. The compression methods are also to help to find the better method in an effective and efficient manner. One of the disadvantage for this comparison is not suitable for all type of compression methods. In, future this work can be evaluated using another all compression methods and find a new effective method.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution.

REFERENCES

[1] Information hiding: a new approach in text steganography , l. y. por, b. delina , faculty of computer science and information technology, university of malaya, 50603, kuala lumpur, malaysia porlip@um.edu.my, delinabeh@yahoo.com.

[2] Image information hiding –an survey, d. saravanan, a. ronald doni& a. abisha ajith sathyabama university, chennai, tamilnadu, india
[3] Information hiding codes and their applications to images and audioby mehmet kivanc¸ mihc¸ akb.s.,bilkent university, 1996 m.s., university of illinois at urbana-champaign, 1999
[4] Information hiding techniques: a tutorial review , sabu m thampi ,assistant professor department of computer science & engineering ,lbs college of engineering, kasaragod kerala- 671542, s.india smtlbs@yahoo.co.in
[5] Ieee journal on selected areas in communications .special issue on copyright and privacy protection, vol. 16, no. 4, may 1998.
[6] Proceedings of the ieee .special issue on identification and protection of multimedia information, vol. 87, no. 7, july 1999.
[7] f. a. p. petitcolas, r. j. anderson, and m. g. kuhn, "information hiding - a survey," proceedings of the ieee .special issue on protection of multimedia content, vol. 87, no. 7, pp. 1062-1078, july 1999.
[8] w. bender, d. gruhl, n. morimoto, and a. liu, "techniques for data hiding," in proceedings of spie , 1995, pp. 2420-2440. [13] i. j. cox, j. killian, f. t. leighton, and t. shamo on, "secure spread spectrum watermarking for multimedia," ieee transactions on image processing , vol. 6, no. 12, pp. 1673-1687, dec. 1997.
[9] f. hartung and m. kutter, "multimedia watermarking techniques," proceedings of the ieee .special issue on protection of multimedia content, vol. 87, no. 7, pp. 1079-1107, july 1999.
[10] m. d. swanson, m. koyabashi, and a. h. tewfik, "multimedia data-embedding and watermarking strategies," proceedings of the ieee , vol. 86, no. 6, pp. 1064-1087, june 1998.
[11] r. b. wolfgang, c. i. po dilchuk, and e. j. delp, "perceptual watermarks for digital images and video," proceedings of the ieee . sp ecial issue on protection of multi- media content, vol. 87, no. 7, pp. 1108-1126, july 1999.
[12] Read M. Saleh, (2001), "Information Hiding in Wave Media File by Using Low Bit Encoding", M.Sc thesis, University of Technology, Baghdad, Iraq.
[13] Provos N., (January 31, 2001) "Probabilistic Methods for Improving Information Hiding", Center for Information Technology Integration, University of Michigan, USA. Email: provos@citi.umich.edu
[14] Luis von Ahn,, Manuel Blum,, Nicholas J.Hopper , and John Langford , (2003),"Using Hard AI Problems For Security", Computer Science Dept., Carnegie Mellon University, Pittsburgh PA 15213, USA .
[15] T.J. Watson Research Center, Yorktown Heights NY 10598,USA. Jacob T. Jackson, Gregg H. Gunsch, Roger L. Claypoole, Jr., and Gary B.
[16] Lamont, (2003)" Novel Steganography Detection Using an Artificial Immune System Approach ", Air Force Institute of Technology Department of Electrical and computer Engineering 2950 Hobson Way, Bldg 640 Wright .
[17] Compressed Video over Networks, Editors Ming-Ting Sun and Amy R. Reibman, Marcel Dekker Publishers, 1st edition, 2001.
[18] R. Rajan, D. Verma, S. Kamat, E. Felstaine, S. Herzog, "A Policy Framework for Integrated and Differentiated Services in the Internet," IEEE Network Magazine, Sept./Oct. 1999.
[19] K. Stuhlmuller, N. Farber, M. Link, B. Girod, "Analysis of Video Transmission over Lossy Channels," IEEE Journal on Selected Areas in Communication, Vol. 18, No 6, June 2000.
[20] M. van der Schaar, S. Krishnamachari, S. Choi, X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs," IEEE Journal on Selected Areas in Communication, Vol. 21, Issue 10, Dec. 2003, pp. 1752.1763.
[21] D. Krishnaswamy, R. Stacey, R. van Alstine, W. Chimitt, "Performance Considerations for Efficient Multimedia Streaming in Wireless Local-Area-Networks," 49th Annual SPIE conference on Applications of Digital Image Processing, August 2004.
[22] D. Krishnaswamy, "Game Theoretic Formulations for network-assisted resource management in wireless networks," IEEE Vehicular Technology Conference, pp. 1312.1316, Sept. 2002.
[23] D. Krishnaswamy, M. van der Schaar, "Adaptive Modulated Scalable Video Transmission over Wireless Networks with a Game-Theoretic Approach," IEEE Multimedia Signal Processing Workshop, September 2004.
[24] B. Pfitzmann, "Information Hiding Terminology", First International Workshop on Information Hiding, May 30 – June 1, 1996, Cambridge, UK, pp. 347-350.
[25] Rade Petrovi, Kanaan Jemili, Joseph M. Winograd, Ilija Stojanovi, Eric Metois, "DATA HIDING WITHIN AUDIO SIGNALS", June 15, 1999, MIT Media Lab, Series: Electronics and Energetics vol. 12, No.2, pp. 103-122.
http://pubs.media.mit.edu/?section=docdetail&id=211474&collection=Media+Lab&filtercollection=Media+Lab
[26] J. Johnston and K. Brandenburg, "Wideband Coding Perceptual Consideration for Speech and Music". Advances in Speech Signal Processing, S. Furoi and M. Sondhi, Eds. New York: Marcel Dekker, 1992.
[27] W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, S. Pogreb, "Techniques for data hiding", IBM Systems Journal, Volume 39 , Issue 3-4, July 2000, pp. 547 – 568.
[28] Samir Kumar Bandyopadhyay, Debnath Bhattacharyya, Poulami Das, Debashis Ganguly and Swarnendu Mukherjee, "A tutorial review on Steganography", International Conference on Contemporary Computing (IC3-2008), Noida, India, August 7-9, 2008, pp. 105-114.

[29] Grib onval, R., Benaroya, L., Vincent, E., and Fevotte, C.: Proposals for performance measurement in source separation. In Pro c. Int. Symp. Independent Component Analysis and Blind Source Separation (ICA'03), April 2003.
[30] Hyvarinen, A. and Karthikesh, R.: Imposing sparsity on the mixing matrix in independent component analysis. Neuro computing, 49, 2002.
[31] 4. Boflll, P. and Zibulevsky, M.: Underdetermined blind source separation using sparse representations. Signal Processing, 81(11), November 2001.
[32] Li, Y., Cichocki, A., and Amari, S.: Analysis of sparse representation and blind source separation. Neural Computation, 16(6), June 2004.
[33] Comon, P.: Independent component analysis, a new concept Signal Processing, 36(3), April 1994.
[34] Hyvarinen, A.: Survey on independent component analysis. Neural Computing Surveys, 2, 1999.
[35] Po or, H.: An Introduction to Signal Detection and Estimation. Berlin, Germany, Springer- Verlag, 1994.

