International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 10, October 2015
Available at http://internationaljournalofresearch.org

Application of Data Hiding in Audio -Video Using Anti Forensics Technique for Authentication and data Security

[1]I Adum Babu & [2]Md Khaleelpasha
[1]associate Professor, Mlritm Marri Laxman Reddy Institute Of Technology & Management
[2]m.Tech (Embedded Systems Marri Laxman Reddy Institute Of Technology & Management)
[1]mail Id: Adumbabu@Gmail.Com [2]mail Id: Khaleel9333@Gmail.Com

Abstract—

LSB Steganography is a method where one can encrypt image/text into an image/Audio for secured data transmission. This paper deals with the detection of hidden bits in the Least Significant Bit (LSB) plane of a natural image /audio. The mean level and the covariance matrix of the image, considered as a quantized Gaussian random matrix, are unknown. An adaptive statistical test is designed such that its probability distribution is always independent of the unknown image parameters, while ensuring a high probability of hidden bits detection. This test is based on the likelihood ratio test except that the unknown parameters are re-placed by estimates based on a local linear regression model. It is shown that this test maximizes the probability of detection as the image size becomes arbitrarily large and the quantization step vanishes. This provides an asymptotic upper-bound for the detection of hidden bits based on the LSB replacement mechanism. System is developed on a Laptop/PC where python-openCV is installed and further data is sent to ARM processor for manipulation.

Keywords— Adaptive detection; information hiding; python-open CV; ARM7 LPC2148; XBEE PRO

I. INTRODUCTION

One of the reasons that intruders can be successful is that most of the information they acquire from a system is in a form that they can read and comprehend. Intruders may reveal the information to others, modify it to misrepresent an individual or organization, or use it to launch an attack. One solution to this problem is to use steganography. Steganography is a technique of hiding information in digital media. In contrast to cryptography, it is not to keep others from knowing the hidden information but it is to keep others from knowing the hidden information but it is to keep others from thinking that the information even exists.

The goal of Steganography is to avoid drawing suspicion to the existence of a hidden message. This approach of information hiding technique has recently become popular in a number of application areas. Digital audio, video, and images are increasingly furnished with distinguishing but imperceptible marks, which may contain a hidden copyright notice or serial number or even help to prevent unauthorized copying directly. Nowadays images are the most popular cover object used for steganography where an altered image with slight variations in its colors will be indistinguishable from the original image by a human being, and thus the importance of Image steganography. In this work, text, image, binary, audio steganography were implemented with ARM Processor and openCV.

II. SYSTEM ARCHITECTURE

The system architecture of this proposed system is given below with simple blocks:

ARM7 END: Hardware implementation for this proposed system is shown below with the simple blocks. Power Supply block is designed and developed to generate power source for the ARM processor and its relevant components. Reset Circuit is designed and developed to reset the program whenever necessary and interfaced to the ARM processor for greater stable response. Clock Circuit is designed and developed to generate oscillations and interfaced to the ARM processor for needy response. LCD Display can also interface to the ARM processor for displaying the status of the system for better understanding. Through XBEE Module, the information is communicated from openCV to ARM processor.

BLOCK DIAGRAM

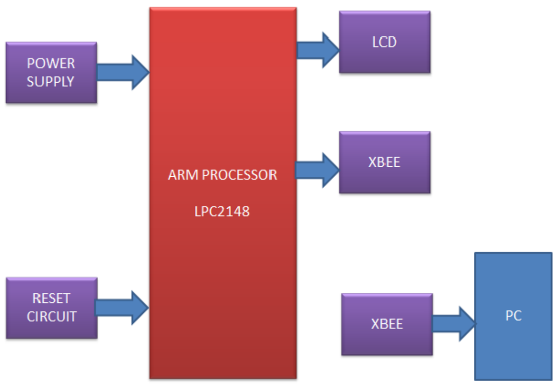


Figure 1 – Block Diagram

III. IMPLEMENTATION

HARDWARE:

ARM7 END:

In hardware implementation, ARM processor plays a key role in monitoring and controlling the security system. Low-power consumption ARM processor (LPC2148) operating at 3.3V, 50uA is designed and mounted on a PCB along with Reset Circuit and a Clock Circuit. LPC2148, a 32-bit microcontroller with advanced RISC architecture and having 48 GPIO lines with a program memory of 32KB and a data memory of 512Bytes.

And we have 2 UART ports i.e. UART0 and UART1. In this project XBEE connected to the UART0 port of ARM7 (LPC 2148). And 3 Analog to Digital channels, though I connected three Analog sensors to ADC channels of ARM7, so that it converts Analog Values to Digital Values. Those values i have uploaded into ThingSpeak.

Each Sensors and its behaviour explained in below. And ARM7 (LPC 2148) internal architecture overview has shown below as well ARM7 (LPC 2148) with LCD has shown below.

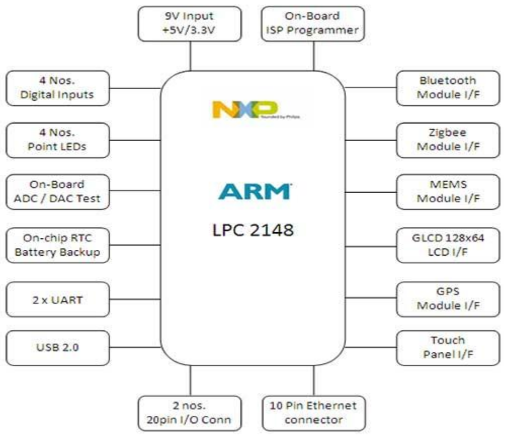


Figure – 2: ARM Overview [LPC2148]

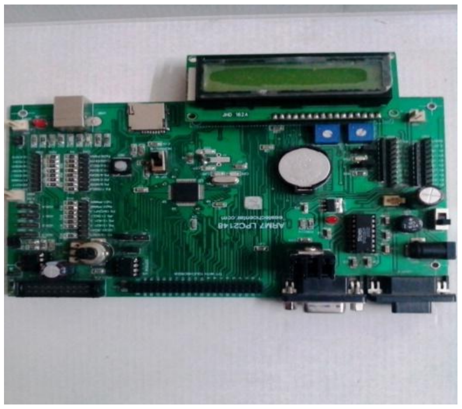


Figure – 3: LPC2148 Development Board

ZIGBEE or XBEE TECHNOLOGY:

Zigbee is an IEEE 802.15.4 standard for data communications with business and consumer devices. It is designed around low-power consumption allowing batteries to essentially last forever. The Zigbee standard provides network, security and application support services operating on top of the IEEE 802.15.4 Medium Access Control (MAC) and Physical Layer (PHY) wireless standard. It employs a suite of technologies to enable scalable, self-organizing, self-healing networks that can manage various data traffic patterns.

The Architecture of zigbee is given below:

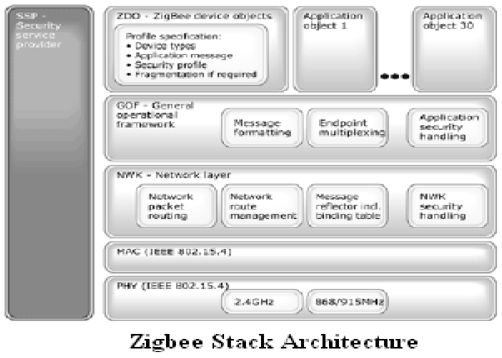


Figure 4 – Zigbee Architecture

Zigbee is a home area network designed specifically to replace the proliferation of individual remote controls. Zigbee was created to satisfy the market's need for a cost effective, standards based wireless network that supports low data rates, low power consumption, security and reliability. To address this need, the ZigBee Alliance, an industry working group (www.zigbee.org), is developing standardized application software on top of the IEEE 802.15.4
wireless standard. The alliance is working closely with the IEEE to ensure an integrated, complete and interoperable network for the market. For example, the working group will provide interoperability certification testing of 802.15.4 systems that include the ZigBee software layer.

The ZigBee Alliance will also serve as the official test and certification group for ZigBee devices. ZigBee is the only standards based technology that addresses the needs of most remote monitoring and control and sensory network applications.

Schematic Diagram of this project has shown below.

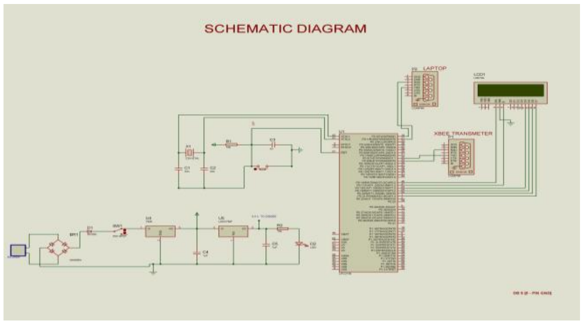


Figure 5 – Schematic Diagram

Power and Reset circuits are connected to the ARM7 (LPC 2148) and LCD has connected as an output to the ARM 7 (LPC 2148) and XBEE has connected to the ARM7 end. And another XBEE connected to the PC.

SOFTWARE:

Here, to program ARM processor Keil uVision 4 was used as a cross-compiler and Flash Magic was used as a programmer.

ALGORITHM&FLOWCHART

FLOWCHART FOR STEGANOGRAPHY:

Usually 24-bit or 8-bit files are used to store digital images. The former one provides more space for information hiding, however, it can be quite large. The colored representations of the pixels are derived from three primary colors: red, green and blue. 24-bit images use 3 bytes for each pixel, where each primary color is represented by 1 byte. Using 24-bit images each pixel can represent 16,777,216 color values. We can use the lower two bits of these color channels to hide data, then the maximum color change in a pixel could be of 64-color values, but this causes so little change that is undetectable for the human vision system.

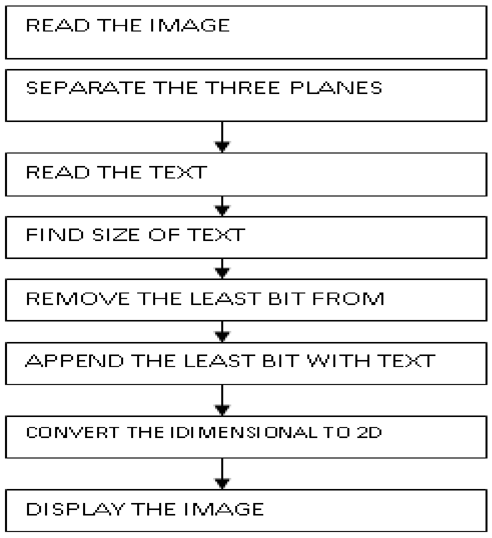
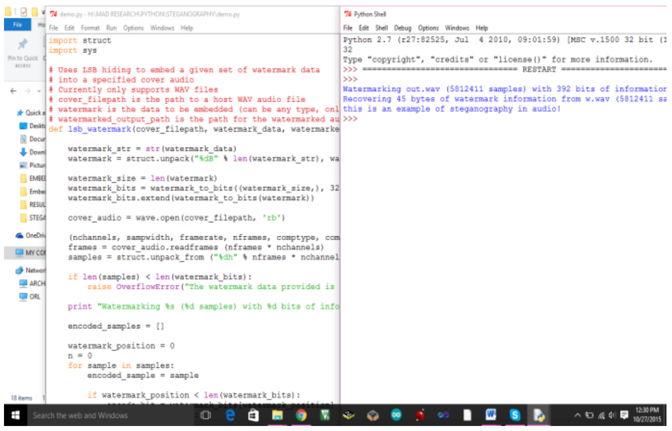

READ THE IMAGE
SEPARATE THE THREE PLANES
READ THE TEXT
FIND SIZE OF TEXT
REMOVE THE LEAST BIT FROM
APPEND THE LEAST BIT WITH TEXT
CONVERT THE IDI MENSIONAL TO 2D
DISPLAY THE IMAGE

Figure 6 – Flow Chart

This simple method is known as Least Significant Bit insertion . Using this method it is possible to embed a significant amount of information with no visible degradation of the cover image. Fig. 2 shows the process.

Several versions of LSB insertion exist. It is possible to use a random number generator initialized with a stego-key and its output is combined with the input data, and this is embedded to a cover image. For example in the presence of an active warden it is not enough to embed a message in a known place (or in a known sequence of bits) because the warden is able to modify these bits, even if he can't decide whether there is a secret message or not, o r he can't read it because it is encrypted. The usage of a stego-key is important, because the security of a protection system should not be based on the secrecy of the algorithm itself, i nstead of the choice of a secret key. Fig. 3 shows this process. The LSB inserting usually operates on bitmap images. 'Steganos for Win dows' and 'Wbstego' are LSB inserting software products which are able to embed data (in clear or encrypted format) in a bitmap image. The embedded data cannot be considered as a watermark, because even if a small change occurs in a picture (cropping, lossy compression, color degradation) the embedded information will be lost – although the change which is occurred during the embedding process is invisible.

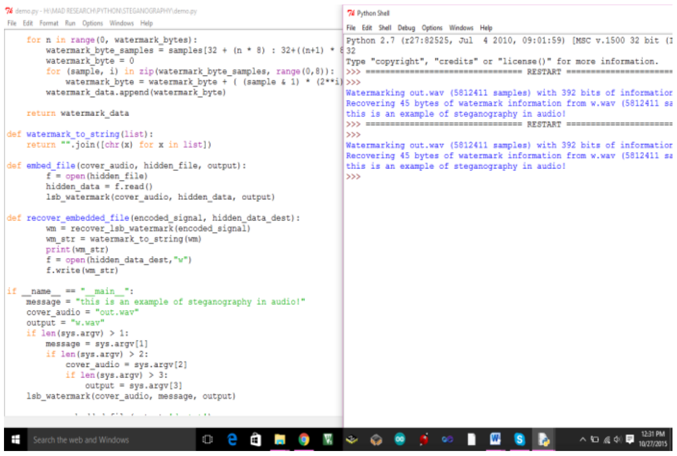
IV. RESULTS



```
import struct
import sys

# Uses LSB hiding to embed a given set of watermark data
# into a specified cover audio
# Currently only supports WAV files
# cover_filepath is the path to a host WAV audio file
# watermark is the data to be embedded (can be any type, onl
# watermarked_output_path is the path for the watermarked au
def lsb_watermark(cover_filepath, watermark_data, watermarke

    watermark_str = str(watermark_data)
    watermark = struct.unpack("%dB" % len(watermark_str), wa

    watermark_size = len(watermark)
    watermark_bits = watermark_to_bits((watermark_size,), 32
    watermark_bits.extend(watermark_to_bits(watermark))

    cover_audio = wave.open(cover_filepath, 'rb')

    (nchannels, sampwidth, framerate, nframes, comptype, com
    frames = cover_audio.readframes (nframes * nchannels)
    samples = struct.unpack_from ("%dh" % nframes * nchannel

    if len(samples) < len(watermark_bits):
        raise OverflowError("The watermark data provided is

    print "Watermarking %s (%d samples) with %d bits of info

    encoded_samples = []

    watermark_position = 0
    n = 0
    for sample in samples:
        encoded_sample = sample

        if watermark_position < len(watermark_bits):
```

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (I
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ========================
>>>
Watermarking out.wav (5812411 samples) with 392 bits of informatio
Recovering 45 bytes of watermark information from w.wav (5812411 sa
this is an example of steganography in audio!
>>>
```

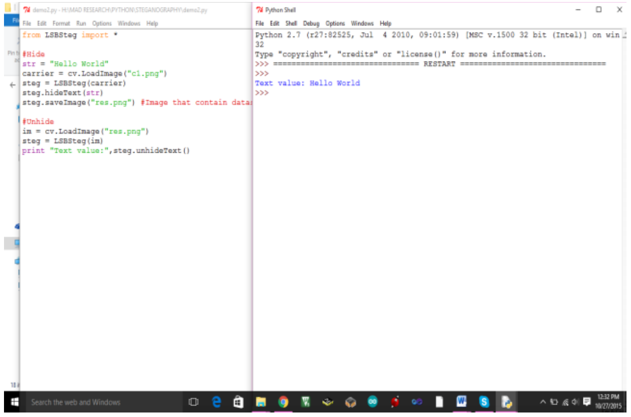
Figure 7 –Final Result 1



```
    for n in range(0, watermark_bytes):
        watermark_byte_samples = samples[32 + (n * 8) : 32+((n+1) * 8
        watermark_byte = 0
        for (sample, i) in zip(watermark_byte_samples, range(0,8)):
            watermark_byte = watermark_byte + ( (sample & 1) * (2**i)
        watermark_data.append(watermark_byte)

    return watermark_data

def watermark_to_string(list):
    return "".join([chr(x) for x in list])

def embed_file(cover_audio, hidden_file, output):
        f = open(hidden_file)
        hidden_data = f.read()
        lsb_watermark(cover_audio, hidden_data, output)

def recover_embedded_file(encoded_signal, hidden_data_dest):
        wm = recover_lsb_watermark(encoded_signal)
        wm_str = watermark_to_string(wm)
        print(wm_str)
        f = open(hidden_data_dest,"w")
        f.write(wm_str)

if __name__ == "__main__":
    message = "this is an example of steganography in audio!"
    cover_audio = "out.wav"
    output = "w.wav"
    if len(sys.argv) > 1:
        message = sys.argv[1]
        if len(sys.argv) > 2:
            cover_audio = sys.argv[2]
            if len(sys.argv) > 3:
                output = sys.argv[3]
    lsb_watermark(cover_audio, message, output)
```

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (I
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ========================
>>>
Watermarking out.wav (5812411 samples) with 392 bits of informatio
Recovering 45 bytes of watermark information from w.wav (5812411 sa
this is an example of steganography in audio!
>>> ================================ RESTART ========================
>>>
Watermarking out.wav (5812411 samples) with 392 bits of informatio
Recovering 45 bytes of watermark information from w.wav (5812411 sa
this is an example of steganography in audio!
>>>
```

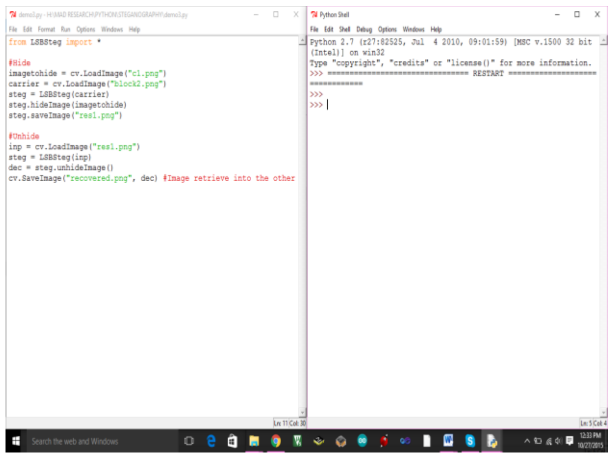
Figure 8– Final Result 2



```
from LSBSteg import *

#Hide
str = "Hello World"
carrier = cv.LoadImage("c1.png")
steg = LSBSteg(carrier)
steg.hideText(str)
steg.saveImage("res.png") #Image that contain data

#Unhide
im = cv.LoadImage("res.png")
steg = LSBSteg(im)
print "Text value:",steg.unhideText()
```

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Text value: Hello World
>>>
```

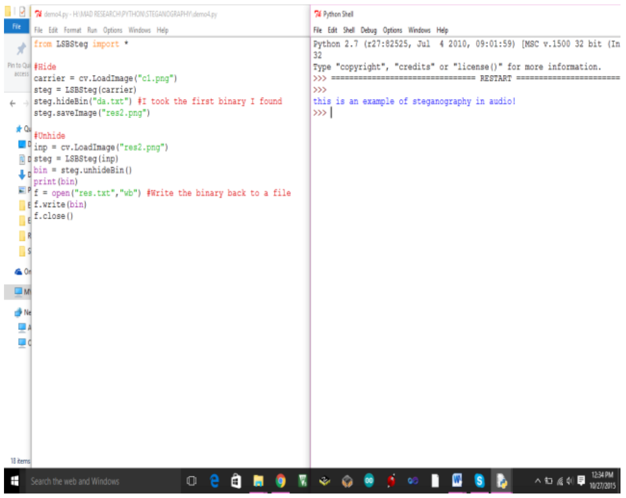
Figure 9 – Final Result 3



```
from LSBSteg import *

#Hide
imagetohide = cv.LoadImage("c1.png")
carrier = cv.LoadImage("block2.png")
steg = LSBSteg(carrier)
steg.hideImage(imagetohide)
steg.saveImage("res1.png")

#Unhide
inp = cv.LoadImage("res1.png")
steg = LSBSteg(inp)
dec = steg.unhideImage()
cv.SaveImage("recovered.png", dec) #Image retrieve into the other
```

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ===================
============
>>>
>>>
```

Figure 10 – Final Result 4


demo4.py - H:\MAD RESEARCH\PYTHON\STEGANOGRAPHY\demo4.py
File Edit Format Run Options Windows Help
Python Shell
File Edit Shell Debug Options Windows Help

```
from LSBSteg import *

#Hide
carrier = cv.LoadImage("c1.png")
steg = LSBSteg(carrier)
steg.hideBin("da.txt") #I took the first binary I found
steg.saveImage("res2.png")

#Unhide
inp = cv.LoadImage("res2.png")
steg = LSBSteg(inp)
bin = steg.unhideBin()
print(bin)
f = open("res.txt","wb") #Write the binary back to a file
f.write(bin)
f.close()
```

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (In
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ======================
>>>
this is an example of steganography in audio!
>>> 
```

Figure 11 – Final Result 5

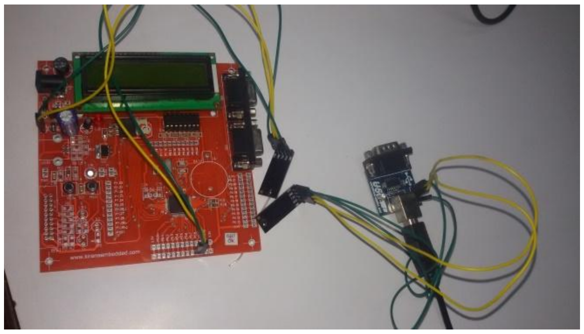


Figure 12 – Final Result 6

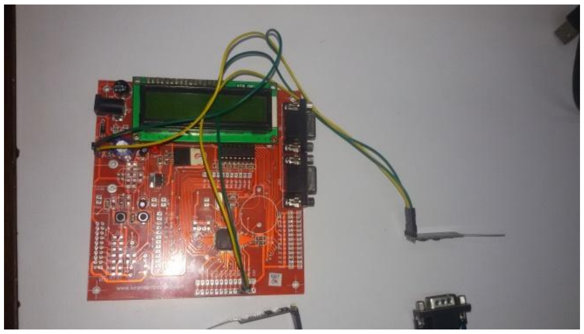


Figure 13 – Final Result 7

V. Conclusion and Futurework

In this Project Image processing system will be implemented on the Laptop/PC and python-openCV.Binary, text, image and Audio Steganography were implemented.

This can be imported to a processor, and can develop some protocols related to steganography.

REFERENCES

[1] H. Sencar, M. Ramkumar, and A. Akansu, Data Hiding Fundamentalsand Applications: Content Security in Digital Multimedia. Elsevier:Academic, 2004.

[2] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker , Digital Water-marking and Steganography. San Francisco, CA: Morgan Kaufmann,2007.

[3] J. Fridrich , Steganography in Digital Media—Principles, Algorithms,and Applications . New York: Cambridge Univ. Press, 2009.

[4] R. Böhme , Advanced Statistical Steganalysis. New York: Springer,2010.

[5] N. Provos and P. Honeyman, “Hide and seek: An introduction tosteganography,” IEEE Secur. Priv. J., vol. 1, no. 3, pp. 32–44, 2003.

[6] X.-Y. Luo, D.-S. Wang, P. Wang, and F.-L. Liu, “A review on blinddetection for image steganography,” Signal Process., vol. 88, no. 9,pp.2138–2157, Sep. 2008.

[7] A. Nissar and A. Mir, “Classification of steganalysis techniques: Astudy,” Digit. Signal Process. , vol. 20, no. 6, pp. 1758–1770, 2010.

[8] Li, S.L., Leung, K.C., Cheng, L.M., Chan, C.K. “Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing” , icicic 2006, IS16-005.

[9] Chang, C.C., Lin, M.H., Hu, Y.-C., 2002, “A fast and secure image hiding scheme based on LSB substitution”, Int. Journal of Pattern Recognit. And Artif.Intell. 16 (4), pp 399-416.

[10] Chan, C.K., Cheng, L.M., 2004, “Hiding data in images by simple LSB substitution”, Pattern Recognition 37, pp469–474.

[11] Wang, H., Wang, S, October 2004, "Cyber warfare: Steganography vs. Steganalysis", Communications of the ACM, 47:10.

[12] Sharp, T., 2001, "An implementation of key-based digital signal steganography", Proc. 4th International Workshop on Information Hiding, Springer LNCS, vol. 2137, pp.13-26.

BIOGRAPHY:

Associate Prof I ADUM BABU, B.TECH from CHIRALA ENGINEERING COLLEGEand M.TECH from ST MARY's COLLEGE of ENGINEERING and TECHNOLOGY(EMBEDDED SYSTEMS)Hyderabad. He is presently working as ASSOCIATE PROFESSOR in MLRITM dundigal.

Mr.MD KHALEELPASHA had completed B.TECH in ECE from MALLA REDDY INSTITUTE OF ENGINEERING & TECHNOLOGYHyderabad.and Pursuing M.TECH In Embedded Systems from MLRITM College Dundigal