International Journal of Information Technology and Knowledge Management
July-December 2009, Volume 2, No. 2, pp. 289-295

PATTERN MATCHING WITH EXTERNAL HARDWARE FOR STEGANOGRAPHY ALGORITHM

Bawankar Chetan D.[*], Hande K. N.[*], Jaiswal A. A.[**] & Bute. A.[**]

Many steganography techniques are extremely vulnerable to attacks and even detection of information in image is quite easy. This paper proposes a new authentication, verification technique, which prevent from the steganographic attacks. The main idea is to use a prioritized sub blocks by pattern matching scheme to embed the code and a microcontroller used for sake of security where it transmits a pre-programmed key at the beginning of each process. The steganalysis algorithm continued only if the received key is correct at the destination otherwise retrieving secure information from cover image is not accomplished.

Keywords: Authentication, Pattern Matching, Steganography Attacks.

I. Introduction

There are many ways to categorize data hiding techniques. A straightforward classification is according to the type of primary multimedia sources, giving us data hiding systems for perceptual and non-perceptual sources. This paper primarily concerned with perceptual multimedia sources, including audio, binary image, color or grayscale image. Among digital sources, the major difference between perceptual and non-perceptual data is that the non-perceptual data, like text and executable codes, usually requires lossless processing, transmission and storage. Flipping a single bit may lead to different meaning. Perceptual data, however, has a perceptual tolerance range, which allows minor change before being noticed by human. This perceptual property enables data embedding as well as lossy compression either imperceptibly or with a controllable amount of perceptual degradation. Although many general techniques of data hiding can be applied to audio, image. There are unique treatments associated with each type of perceptual sources.

The digital information revolution has brought about profound changes in our society and our lives. The many advantages of digital information have also generated new challenges and new opportunities for innovation. The issues regard multimedia data hiding and its application to multimedia security and communication, addressing both theoretical and practical aspects, and tackling both design and attack problems.

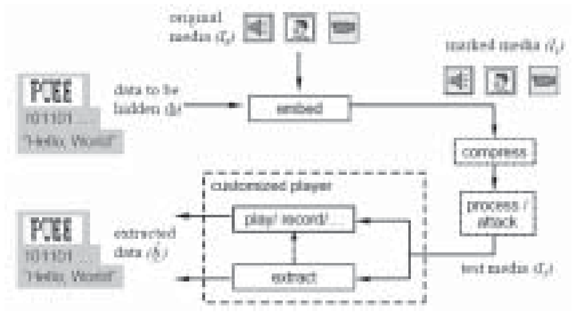

Figure 1: General Frame Work for Data Hiding

The rest of paper is organized as follows:

Section II–Fundamental Issues. Section III–the microcontroller which provide the external security to there algorithm. Section IV–Experimental Result & Finally, Conclusions are drawn in Section V.

II. Fundamental Issues

In the fundamental part, author identify a few key elements of data hiding through a layered structure. Data hiding is modeled as a communication problem where the embedded data is the signal to be transmitted. In addition, author have observed that the unevenly distributed embedding capacity brings difficulty in data hiding. [1, 2]

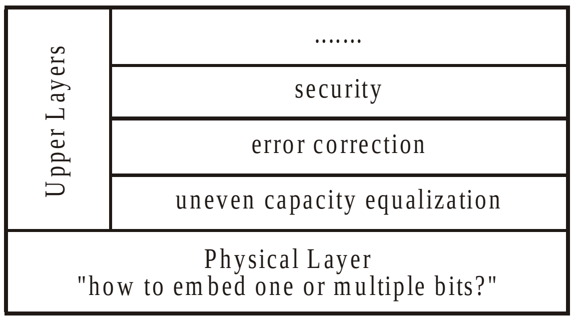
Upper Layers
....
security
error correction
uneven capacity equalization
Physical Layer
"how to embed one or multiple bits?"

Figure 2: Layered Structure

In addition, a layered structure helps to prioritize and compartmentalize various design issues.

[*] Dept. of CSE, G. H. Raisoni College of Engg. Nagpur, India
[**] Dept. of CT, KDK College of Engg. Nagpur, India

The key elements in many data hiding systems include:

1. How to embed one bit.
2. How to embed multiple bits via modulation/ multiplexing techniques.
3. How to embed in those parts of cover media which are difficult to embed, or more generally, how to handle uneven embedding capacity.
4. How to enhance robustness and security.
5. What data to embed.

In this paper, Authors proposed a new algorithm based on symmetric model approach. It uses the text message itself as key for encrypting the data. This algorithm does not restrict only to 26 alphabets but may also consider any special character if any is there. The Binary 4 fold cipher algorithm uses the concept of dividing the given text in two equal length sub string & again apply dividing logic on both sub string & then by applying the encryption logic among these substring to get the cipher text. A Secure Authentication Technique for binary images using Pattern Matching is also proposed by authors. The approach can be used to verify whether a binary document has been tampered with or not and also authenticates the originator. The original image is represented as *F*. It is partitioned into *m* × *n* sub blocks (say 3 × 3). In each sub block only the middle pixel is used to hide the information. Each 3 × 3 sub-block is checked against predefined patterns. If a sub-block matches with any of the valid patterns, it is ready to hide the information. [6, 7]

Binary 4 Fold Cipher Algorithm:

Securing the text is the prime need of today's era. The suggested algorithm provides a simple way for encrypting the text. Text may comprise of alphabets or any another special characters. The author emphasis on the symmetric approach of encryption.

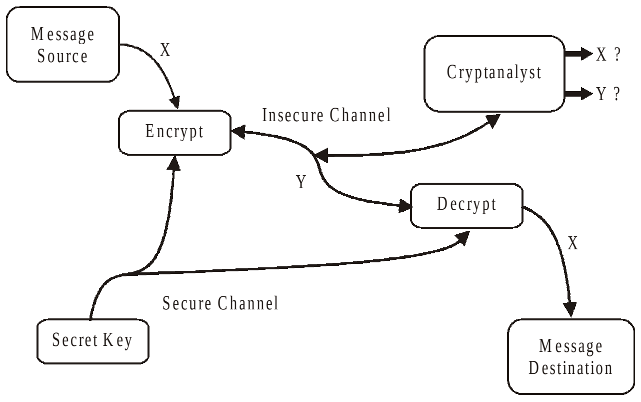

Figure 3: Symmetric Approach of Encryption

Encryption Algorithm:

1. Read the message (*S*)
2. Get the length of message (*n*).
3. Check the value of *n*, if *n* is even divide it by 2. Separate out two sub matrix *A* & *B* of length 0 to (*n*/2)–1 & (*n*/2) to *n*–1 respectively.
4. Now again check the value of matrix *A* & *B* i.e. *n*, if *n* is even divide it by 2. Separate out two sub matrix *A*1 & *A*2, *B*1 & *B*2.
5. Interchange every even position letter of *A*1 with even position letter of *A*2 & retain the odd position letter as it is in *A*1 as well as *A*2. Join the sub matrix *A*1 & *A*2 to get the Sub cipher text *A*. Repeat for the *B*1 & *B*2 matrix to get Sub cipher text *B*.
6. Join the sub Cipher text *A* & Sub Cipher text *B* to get final Cipher text.
7. If the length is odd then leave the *n*th letter as it is in *S* & consider the text size for encryption as *n*–1 then apply the algorithm specified in step 3. Join *A* & *B* Matrix with *S* to get the cipher text.

Decryption Algorithm:

1. Read the cipher text.
2. Get the length of cipher text (*n*).
3. Check the value of *n*, if *n* is even divide it by 2. Separate out two sub matrix *A* & *B* of length 0 to (*n*/2)–1 & (*n*/2) to *n*–1 respectively.
4. Now again check the value of matrix *A* & *B* i.e. *n*, if *n* is even divide it by 2. Separate out two sub matrix *A*1 & *A*2, *B*1 & *B*2.
5. Interchange every even position letter of *A*1 with even position letter of *A*2 & retain the odd position letter as it is in *A*1 as well as *A*2. Join the sub matrix *A*1 & *A*2 to get the Sub cipher text *A*. Repeat for the *B*1 & *B*2 matrix to get Sub cipher text *B*.
6. Join the sub Cipher text *A* & Sub Cipher text *B* to get final Cipher text.
7. If the length is odd then leave the *n*th letter as it is in *S* & consider the text size for encryption as *n*–1 then apply the algorithm specified in step 3. Join *A* & *B* Matrix with *S* to get the cipher text.

SECURE AUTHENTICATION TECHNIQUE

The original image is represented as *F*. It is partitioned into *m* × *n* sub blocks (say 3 × 3). In each sub block only the middle pixel is used to hide the information. Each 3 × 3 sub-block is checked against predefined patterns. If a sub-block matches with any of the valid patterns, it is ready to hide the information. It is referred as R*eadyBlock*, and the middle pixel of it can be used to hide the information.

But, the idea of computing an AS of the whole binary image and inserting it into the same image fails. Since the insertion will modify the image fingerprint, it can not be verified at the receiver side. A modified idea to insert the AS without modifying the image fingerprint is to divide the image into two regions: The first region is small in size, called AS Region (ASR) where AS will be inserted and the second region is the original image excluding ASR, called Non-AS Region (NASR) from where AS will be computed. Using this technique, the image authenticity can be verified at the receiver side [7, 8, 9].

The SA-PM

In SA-PM, only a few pixels are modified and the positions of sub-locks containing those pixels are known both in the insertion and extraction phases. SA-PM flips only low-visibility pixels to hide the information and consequently the stegno image has excellent visual quality and do not have salt-and-pepper noise.

Finding *ReadyBlocks*

The original image is represented as F. It is partitioned into $m \times n$ sub-blocks Fi. In each sub-block only the middle pixel is used to hide the information. But, not all the sub-blocks are used for hiding the information. Among the 512, 3 × 3 sub-blocks several blocks were rejected due to various reasons like, high visibility, nonreversible at receiver side etc. Only 120 patterns are considered to be valid for data hiding. The Figure 4 depicts the valid patterns. The hatched middle pixel may either be black or white. The change of middle pixel is less noticeable. Mirrors, transposes and reverses of the patterns are also used for hiding the information.

Each 3 × 3 sub-block is checked against various constraints to identify that whether it matches with any of the valid 120 patterns or not. If a sub-block matches with any of the valid patterns, it is a R*eadyBlock.* The patterns are selected based on the following constraints:

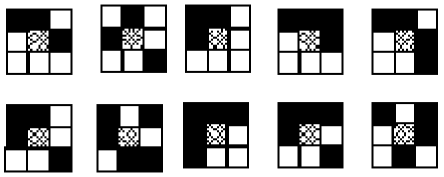

Figure 4: The 3 × 3 Patterns Used for Hiding the Information. The Hatched Middle Pixel may either be Black or White.

The change of middle pixel is less noticeable. Mirrors, transposes and reverses of the above patterns are also used for hiding the information. The remaining patterns are not used.

(a) The number of white pixels in a sub-block should be greater than 2 and less than 6 excluding the middle pixel. That is, 3, 4, 5 or 6 white pixels and 6, 5, 4 or 3 black pixels respectively.

$I1 : X1 < \text{SUM}(Fi) < X2$, $i = 1$ to n; $X1 = 2$;

$X2 = 6$ where,

Fi–a sub-block,

n–total number of sub-blocks

SUM–a function that counts the number of ones (white pixels) in the sub- block.

$X1$ and $X2$–minimum and maximum number of ones in the sub-block respectively.

(b) If there are only 3 white/black pixels, they should not be in same row/column.

If a sub-block satisfies the above constraints, it will become R*eadyBlock*, and ready to hide the information.

Authentication Signature(AS)

Let k be the length of the adopted AS. To insert k bits of AS, it needs k, $m \times n$ *ReadyBlocks* in the image. The image is divided into two regions: The first region is small in size, called AS Region (ASR) where AS will be inserted and the second region is the original image excluding ASR, called Non-AS Region (NASR) from where AS will be computed. Using this technique, the image authenticity can be verified at the receiver side.

Once the *ReadyBlocks* are found, it is clearly understood that the middle pixels of all the *ReadyBlocks* forms the ASR. So, before calculating the hash value of the image, the ASR is made to be darkened. The new image is referred as (ZASR –Zeros inserted at AS Region). The ZASR is actually the whole image with zeros in ASR. Now, the hash value is calculated for ZASR, encrypted using the secret-or public key ks, obtaining the MAC/DS and is inserted in ASR.

At the receiving side, the receiver uses the same technique to identify the R*eadyBlocks*. ASR is separated, encrypted AS inserted in that are retrieved and decrypted using the secret- or public key *ks*, obtaining the AS. Then in the received original image, middle pixels of ASR are made to be darkened. It is referred as ZASR*. Compute H (ZASR*) and if H (ZASR*) = AS, then image integrity is verified. Otherwise, image has been modified or a wrong key was used.

Data Hiding Scheme

This technique ensures that for any pixel that is modified in the host image, its visibility is very less. Thus, the existence of secret information in the host image is difficult to detect. The data hiding scheme is very simple. In each block, the position of the pixel to be used for hiding the information is fixed, that is, the middle pixel. Hence the complexity is not in identifying the position, but in identifying the sub-blocks that are ready to hide the information *(ReadyBlocks)*.

Since the middle pixel is used to hide the information always, another level of security is introduced by shuffling of the *ReadyBlocks.* This provides another advantage also. It distributes the hidden information in all parts of the image. It is an efficient and effective tool to equalize uneven embedding capacity. The *ReadyBlocks* are shuffled and permuted order is generated randomly. A secret key, called *Shuffling Key* shared by the sender and the receiver, is used as the seed for pseudo random number generator. Let the original image F is partitioned into $m \times n$ subblocks *Fi*.

Step 1: If *Fi* is completely black or blank, simply keep *Fi* intact (not hidden with any information) and skip the following steps. Otherwise, perform the following:

Step 2: Identify the *ReadyBlocks* by comparing each subblock *Fi* with the predefined patterns specified in Section Finding *ReadyBlocks.*

Step 3: Use the secret key to generate the permuted order of *ReadyBlocks.*

Step 4: If the middle bit of the *ReadyBlock* and the bit to be hidden are same, then no change is made to the sub-block *Fi*; otherwise, the middle bit is flipped to store the information.

The SA-PM Insertion Algorithm is:

1. Let *F* be a binary image to be watermarked and *k* be the length of AS.
2. Partition the image *F* into $m \times n$ size sub- blocks *Fi*. To insert *k* bits of AS, it needs k, $m \times n$ blocks in the image.
3. Find the *ReadyBlocks* by comparing each subblock *Fi* with the predefined patterns specified.
4. The middle pixels of *ReadyBlocks* forms AS Region. Clear ASR, obtaining ZASR.
5. Compute the hash vale $H = H$ (ZASR).
6. Encrypt *H* using the secret-or public key *ks*, obtaining the digital signature *S* (MAC/DS).
7. Use the secret key to generate the permuted order of *ReadyBlocks*.
8. Insert *S* into ASR.

The SA-PM Extraction Algorithm is :

1. Let F^* be the watermarked image received.
2. Partition the image *F* into $m \times n$ size sub-blocks.
3. Find the *ReadyBlocks* by comparing each subblock *Fi* with the predefined patterns specified.
4. The middle pixels of *ReadyBlocks* forms AS Region. Clear ASR, obtaining ZASR*.
5. Compute the hash vale $H^* = H$(ZASR*).
6. Use the secret key to generate the permuted order of *ReadyBlocks*.
7. Extract the watermark from the *ReadyBlocks* of F^* and decrypt the result using the secret-or public key *ks*, obtaining the digital signature S^*.
8. If S^* and H^* are equal the watermark is verified. Otherwise, the marked image F^* has been modified.

III. External Hardware for Security

In the well-known “prisoners’ problem”, a representative example of steganography, two persons attempt to communicate covertly without alerting the warden. One approach to achieve this task is to embed the message in an innocent-looking cover-media. In our model, the message contents are scattered in the cover in a certain way that is based on a secret key known only to the sender and receiver. Therefore, even if the warden discovers the existence of the message, he will not be able to recover it. In other words a covert or subliminal communication channel is opened between two persons who possess a secret key to reassemble its contents. In this paper, authors propose a steganographic model in which the hidden message can be composed and inserted in the cover in real-time. This is realized by designing and implementing a secret key steganographic microarchitecture employing Field Programmable Gate Arrays FPGA. [2]

In order to evaluate the perceptibility of the steganograpy, subjective tests or a quality metric can be used. The most common use of quality metric for distortion measures in the field of image and video coding and compression are the signal-to-noise ratio (SNR), and the peak signal-to-noise ratio (PSNR). It may be useful to use those distortion metric adapted to the human visual and auditory system. There are four steganographic algorithms selected for evaluation on its suitability in hardware implementation. They are:

1. A image hiding scheme based on pattern matching.
2. Data Hiding in Images by Adaptive LSB Substitution Based on the Pixel-Value Differencing.
3. Steganographic based on predictive neighbor pixels.
4. Simple 2-bit LSB substitution steganography.

In Scheme 1 evaluation test, both of the host image and secret image are 8 bit-grayscale image. The host image size is 512 × 512 and the secret image size is 256 × 256. The peak signal-to-noise ratio (PSNR) is employed to evaluate the stego-image quality. For an *n*-bit image of size *M* × *M* pixels, the PSNR value is defined as follows:

$$PSNR = 10 \times \log_{10} \frac{(2^n - 1)^2}{I / M \times \sum_{i=0}^{M} \sum_{j=0}^{M} (p_g - p_g)^2} \quad (1)$$

Where *ij p* & are the pixel and mean pixel values.

The decoding and encoding timing for this scheme is given in Table 1. The result shows that this steganographic algorithm used a comparatively long time to run, but the quality of the stego-images obtained are quite good. And the mean PSNR value is 44.094 dB, which means the stego-images have a small distortion.

In Scheme 2 algorithm evaluation test, both of the host image and secret-image are 8 bit-grayscale image. The host image size is 512 × 512 and the secret image size is 128 × 128.

Since the embedding capacity of this algorithm is not as good as the algorithm of scheme 1, the image size of the secret image needs to be much smaller. Otherwise, it cannot extract the secret image correctly.

The result also shows that this scheme used a short time to run due to the smaller secret-image size and the quality of the stego-images obtained is also good. And the mean PSNR values are about 42.2062 dB, which is slightly lower than the first scheme.

For the other two schemes, i.e. scheme 3 and 4, the embedding capacity are generally small to maintain a comparable PSNR as in schemes 1 and 2. However, to choose the right schemes for hardware implementation, we have to consider the timing requirement and also the complexity in implementation. A summary of the comparison of different schemes is given in Table 1.

Table 1
A Summary of Timing and Complexity

SCm	*Timing Requirements*	*Complexity in Implementation*	*Advantages*	*Disadvantages*	*Hardware Implementation*
1.	Long	Computation frame by frame & need large buffer	High capacity Stego image	Long processing time	Suitable
2.	Medium	Computation line by line	Short computation time	Required several line buffer	Suitable
3.	Short	Computation using three pixel	Short processing time	Not stable	Suitable
4.	Short	No memory requirements	Short processing time	Small capacity	Suitable

Thus, for Scheme 1, since the algorithm of a novel image hiding scheme based on block difference need to compute a lot of matching processes for each secretimage block with the cover-image block and compute the block header, the processing time of stego-image will be longer. Moreover, this algorithm requires a great deal of memory to store the whole secret-image pixel values and the whole cover-image values for matching; it is not desirable to implement this algorithm in hardware.

For Scheme 2, the memory requirement and the timing requirement of this algorithm are not strict. It only requires several line buffers to store two line pixels value for stego-image. Therefore, it can be developed for hardware to implement.

For Scheme 3, the memory requirement and the timing requirement of this algorithm are also not strict. It requires three pixels buffers for steganographic process. This algorithm is suitable to implement in the hardware.

For Scheme 4, it is the simplest algorithm amongst the selected algorithms. It replaces the 2 LSB of the pixel value with the secret data. This algorithm is easier to implement in hardware.

IV. Experimental Results

Binary 4 Fold Cipher

Encryption Algorithm:

Plain Text : IEEE International Advance Computing Conference 2009.

Cipher Text: iEnElIAtvrnaeIoEandeanctnCemeuci g0C9 forpnten2 0o.

Decryption Algorithm:

Cipher Text : iEnElIAtvrnaeIoEa ndeanctnCemeuci g0C9 forpnten2 0o.

Plain Text : IEEE International Advance Computing Conference 2009.

SECURE AUTHENTICATION TECHNIQUE

An experimental results are illustrated to demonstrate the validity of secure authentication watermarking for binary images using pattern matching. data hiding scheme. Various binary images and document images, such as Mickey Mouse, English text are taken as the sample images in the experiment. The experimental results reveal that the secure authentication for binary images using pattern matching. Data hiding scheme exhibits very good performance.

It shows the original image, Stego image and modified pixels in the stego image of various samples taken. By observing the stego image, it is clear that the visual quality generated by secure authentication for binary images using pattern matching data hiding scheme giving better result.

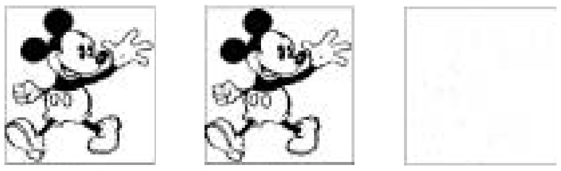

Figure 5: a) Original Copy (b) Marked Copy with 160 bit Embedded in (c) Difference between Original and Marked Image

Distortion Measure

Two of the metrics used to compare the marked image and the original image are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the compressed and the original image, whereas PSNR is a measure of the peak error. It uses a standard mathematical model to measure an objective difference between two images. The mathematical formulae for the two are,

$$MSE = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\frac{(I(x,y)-I'(x,y))2}{M\times N}$$

$$PSNR = 20\times\log 10\frac{255}{sqrt(MSE)}$$

where $I(x, y)$ is the original host image and $I'(x, y)$ is the watermarked image and M, N are the dimensions of the images. A lower value for MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR. Logically, a higher value of PSNR is good because it means that the ratio of Signal to Noise is higher. Here, the 'signal' is the original image, and the 'noise' is the modified pixels in watermarked image. So, if a data embedding scheme having a lower MSE (and a high PSNR), it is recognized as a better one.

The PSNR and MSE obtained for the Mickey mouse image Image size in bits 515 × 512, total no. of $M \times N$ 261244, no. of ready block 330, on. Of pixels modified 89, PSNR in db 29.0226, MSE 0.0013.

Table 2
Encoding and Decoding Timing & PSNR Value for SA PM Algorithm

Host Image (512 × 512)	*Secret Image (256 × 256)*	*Timing for Encoding*	*Timing for Decoding*	*PSNR Value*
Mickey Mouse	Toys	27.5 s	28.6 s	29.0226 dB
Cameraman	Jet	35.99 s	35.13 s	30.286 dB
Pepper	Baboon	27.6 s	38.4 s	33.29 dB
Lena	Toys	28.4 s	42.3 s	35.56 dB

EXTERNAL HARDWARE FOR SECURITY

Table 3
Encoding and Decoding Timing for Scheme 1 Algorithm

Host Image (512 × 512)	*Secret Image (256 × 256)*	*Timing for Encoding*	*Timing for Decoding*	*PSNR Value*
Cameraman	Jet	35.99 s	11.3 s	29.286 dB
Pepper	Baboon	27.6 s	38.4 s	27.29 dB
Lena	Toys	28.4 s	42.3 s	28.56 dB

Table 4
Encoding and Decoding Timing for Scheme 2 Algorithm

Host Image (512 × 512)	*Secret Image (128 × 128)*	*Timing for Encoding*	*Timing for Decoding*	*PSNR Value*
Cameraman	Jet	3.9 s	13.13 s	41.86 dB
Pepper	Baboon	3.6 s	15.4 s	44.59 dB
Lena	Toys	2.9 s	16.3 s	46.56 dB

V. CONCLUSION

Steganography can provide all the aforementioned entities with increased protection for their information. Here the hardware kit with microcontroller gives external security for steganize and de-steganize algorithms. So once steganize the secure information into a cover media using this hardware kit, then to de-steganize the same cover media at the destination, the user must have the similar hardware kit to retrieve the secure information and the user must know the two passwords one which is stored in the microcontroller and other one used during steganography. User without having this hardware kit cant proceed to steganography application and hence gives better security not only to the information hiding but also for steganogarphy algorithm.

REFERENCES

[1] G. J. Simmons. The Prisoner's Problem and the Subliminal Channel. In Advances in Cryptology CRYPTO '83., 1983.

[2] Miroslav Dobsicck "Modern Steganography" In Advances in Cryptology CRYPTO '2005.

[3] Jozsef Lenti "Steganographic Method" Periodica Polytechnica Ser. El. Engg. **44**(3) (2000), 4 249-255.

[4] Bret Dunbar "A Detailed Look at Steganography Tech. and their Use in Open System" at SANS Institute 2002.

[5] W. Bender, D. Gruhi, A. Lu "Techniques for Data Hiding" *IBM System Journal*, **35**(3 & 4) (1996).

[6] Yu-Chee Tseng "A Secure Data Hiding Scheme for Binary Image" *IEEE Journal*, **34** (2002) 0090-6778.

[7] Hac Young Kim "A Public Key Authentication Watermarking for Binary Image" *IEEE Conf. Image Processing*, 0-7803-8554-03/04 (2004).

[8] Mingqiao Wu "Digital Image Steganography Algorithm based on Iterative Blending" *IEEE International Symposium on Signal Processing and Information Tech.* 2005.

[9] S. Voloshynovskiy, S. Pereira, T. Pun, University of Geneva J. J. Eggers and J. K. Su, University of Erlangen-Nuremberg "Attacks on Digital Watermarks: Classification, Estimation-based Attacks and Benchmarks" Submitted to *IEEE Trans.* (2004).

[10] Jessica Fridrich" Attacking the OutGuess" *ACM Conference* '00, Month 1-2, 2000.

[11] J. J. Chae and B. S. Manjunath" A Technique for Image Data Hiding and Reconstruction without Host Image" *NSF* (2000).

[12] Viktor K. Prasanna and Andreas Dandalis" FPGA-based Cryptography for Internet Security" Research Paper Part of the *MAARCII* Project 2004.