DATA EMBEDDING APPROACH FOR MESSAGE AUTHENTICATION SYSTEM

[1]SU MON THU, [2]SU WAI PHYO

[1,2]Department of Computer Engineering and Information Technology, Mandalay Technological University, Mandalay, The Republic of the Union of Myanmar
E-mail: [1]susu052@gmail.com, [2]swp2006@gmail.com

Abstract— The demand for effective information security schemes is increasing day by day with the exponential growth of internet. Therefore, cryptographic algorithms are widely used in many research areas to fulfill the security requirements. Among the security requirements, message authentication plays an important role to prevent message alternation and modification from unauthorized party. Due to attack made on Today's data communication, effective steganographic techniques are also developed using various carriers such as image, audio, video, etc. So, this work proposes security system by integrating cryptographic one-way hash function (SHA-512) and image steganographic approach to meet confidentiality and message authentication requirements. The basic idea behind this paper is to provide a good embedding method based on Pixel Mapping Method (PMM) for data hiding in image to achieve better embedding capacity with minimum degradation of image quality. The performance analysis of proposed embedding approach and PMM is made according to the comparative study of embedding capacity and Peak Signal to Noise Ratio (PSNR) values.

Keywords— Message Authentication, Confidentiality, SHA-512, PMM, PSNR.

I. INTRODUCTION

Data integrity is also important as information security in real world's communication channels. Only hash function can give data integrity and message authentication. Hash function is the most versatile cryptographic algorithm. It is used in a wide variety of security application and internet protocols [1].

Hash function is a technique used to check the authenticity of the data. The two most popular families of protocols to generate the hash function are Secure Hash Algorithm (SHA) and Message Digest (MD). With the help of hash function, the receiver can ensure that the received data is original and it has not been altered while being transported. The generated hash code can be sent to the receiver securely by hiding it into another media such as an image, text, audio or video using steganography [2].
Steganography and Cryptography are two popular ways of sending vital information in a secret way. One hides the existence of the message and the other distorts the message itself. This research paper mainly focuses on to develop a new system with extra security features where a meaningful piece of text message can be hidden by combining security techniques like Cryptography and Steganogrphy. The main advantage of the system is that it encompasses methods of transmitting secret messages through innocuous cover carriers in such a manner that the very existence of the embedded messages is undetectable.
Image steganography is the most popular of the lot. In this method, the secret message is embedded into an image as noise to it, which is nearly impossible to differentiate by human eyes [3], [4], [5]. In this paper, SHA-512 based message authentication system is integrated with data embedding approach to develop better security system.
Moreover, data embedding approach is focused on the optimization of PMM to achieve powerful embedding method in image steganography according to better embedding capacity with less distortion of image quality.
The rest of the paper is organized as follows: section 2 presents the proposed system model. Results and discussion are presented in section 3. Finally, the conclusion of this research work is drawn.

II. THE PROPOSED SYSTEM MODEL

2.1. Proposed System Architecture

The proposed system is composed of two portions: sender and receiver. In **Fig.1**, the original message (M) and common secret value (S) are concatenated to form a hash code H (M || S) with the help of SHA-512 hash algorithm. And then, M and H (M || S) are concatenated to embed into a cover image through OPMM embedding algorithm to get stego-image.

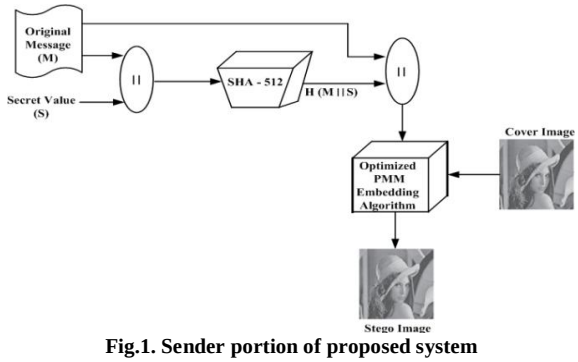

Fig.1. Sender portion of proposed system

The receiver side of the proposed system is illustrated in **Fig.2**. At first, the receiver extracts the

concatenated value with the help of OPMM extraction algorithm. Secondly, the extracted value is split into the original message (M) and hash code H (M || S). To check the message integrity, the message (M) and secret value (S) are concatenated to calculate the hash code H' (M || S) by using SHA-512 hash function. Finally, the resulting two hash messages H (M || S) and H' (M || S) are compared whether it is identical or not. If it is exactly the same, it ensures the data integrity.

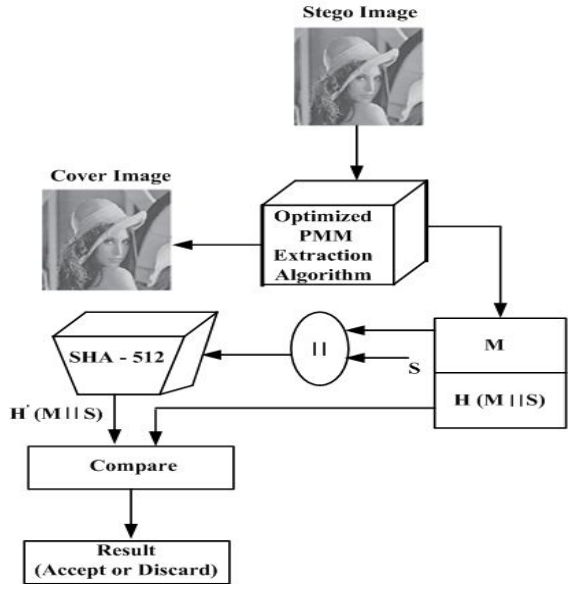

Fig.2. Receiver portion of proposed system

2.2. SHA-512 Logic

The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. **Fig.3** depicts the overall processing of a message to produce a digest [6].

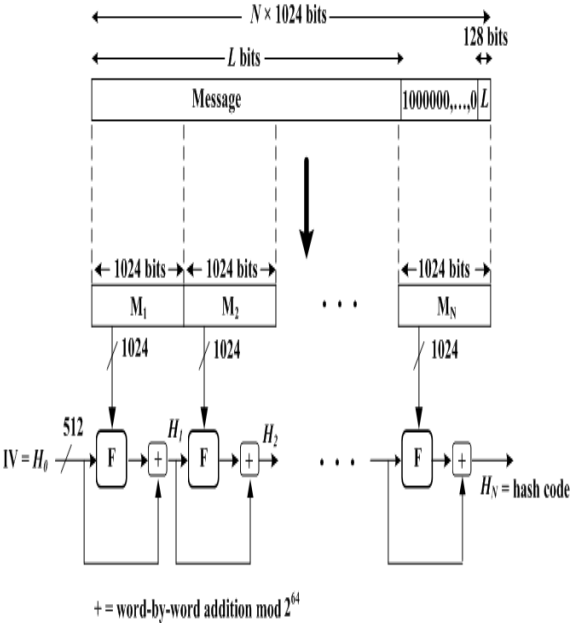

Fig.3. Message digest generation using SHA-512

The processing consists of the following steps:

Step 1: Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024 [length 896 (mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

Step 3: Initialize hash buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).

Step 4: Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in **Fig.3**.

Step 5: Output. After all *N* 1024-bit blocks have been processed, the output from the *N*th stage is the 512-bit message digest [6].

2.3. Pixel Mapping Method (PMM)

PMM is a method for information hiding within the spatial domain of any gray scale image. The input messages can be in any digital form, and are often treated as a bit stream. Embedding pixels are selected based on some mathematical function which depends on the pixel intensity value of the seed pixel and its 8 neighbors are selected in counter clockwise direction. Before embedding a checking has been done to find out whether the selected embedding pixels or its neighbors lies at the boundary of the image or not.

Random pixel generation for embedding message bits is dependent on the intensity value of the previous pixel selected. It includes a decision factor (dp) which is dependent on intensity with a fixed way of calculating the next pixel. The algorithm for selection of pixel for embedding is described below:

Input: C, previous pixel position (x, y), pixel intensity value (v).
Consider dp (Decision Factor) = 1 if (intensity $\leq$ 80), dp = 2 if (intensity $>$ 80 & $\leq$ 160), dp = 3 if (intensity $>$160 & $\leq$ 255).
t = x + 2 + dp
if **(** t $\geq$ N **)** m = 2, n = y + 2 + dp
else m = x + 2 + dp, n = y
Return m and n.
End

Data embedding is done by mapping each four bits of the secret message in each of the neighbor pixel based on some features of that pixel. **Fig.4** shows the mapping information for embedding four bits [7].

Msg Bit Seq	2nd Set-Reset Bit	3rd Set-Reset Bit	Pixel Intensity Value	No of Ones (Bin)
0000	even	even	even	even
0001	even	even	even	odd
0010	even	even	odd	even
0011	even	even	odd	odd
0100	even	odd	even	even
0101	even	odd	even	odd
0110	even	odd	odd	even
0111	even	odd	odd	odd
1000	odd	even	even	even
1001	odd	even	even	odd
1010	odd	even	odd	even
1011	odd	even	odd	odd
1100	odd	odd	even	even
1101	odd	odd	even	odd
1110	odd	odd	odd	even
1111	odd	odd	odd	odd

Fig.4. Mapping technique for embedding of four bits

Extraction process starts again by selecting the same pixels required during embedding. At the receiver side, other different reverse operations have been carried out to get back the original information [7].

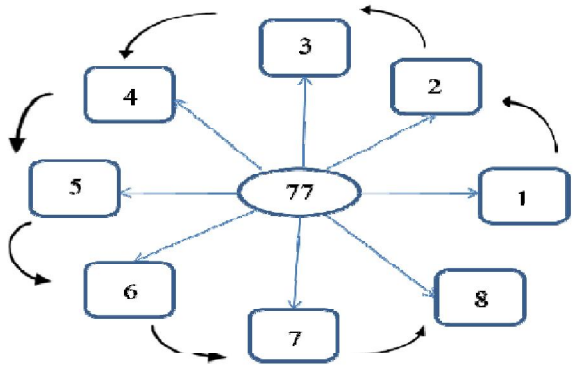

Fig.5. Sequence of data embedding

One important point needs to be kept in mind that a specific order for selecting the neighbors of the seed pixel has to be maintained for embedding/mapping process and also for the process of extraction otherwise it would not be possible to retrieve the data in proper sequence. This sequence has been shown in **Fig.5** [7].

2.4. Optimized Pixel Mapping Method (OPMM)

The optimized pixel mapping method (OPMM) is based on PMM. Like PMM, optimized PMM depends on the pixel intensity value to find the seed pixel and its 8 neighbors are also selected in counter clockwise direction. In OPMM, it is divided into same size blocks including seed pixel and its eight neighbors. Unlike PMM, the continuous nine blocks are vertically selected at a time. Then, the very first block is picked up as a key block and the embedding positions are marked in it. For the secret message embedding, the starting pixel is considered according to the number of ones in the binary of seed pixel. The continuous remaining eight blocks are planned for message embedding. The detail description of OPMM is presented in following pixel selection and embedding algorithms.

2.4.1. Pixel Selection Algorithm of OPMM

Input: Bitmap Cover Image (img), Pixel Position (x,y), Pixel Intensity Value (V)

1. Declare integer variables blockPixels, numPixels, numBlocks, keyStartPixel, msgStartPixel, currBlock, msgBit, msgBlock, difference
2. Set difference to 0
3. Set blockPixels to 9
4. width = img.getWidth
5. height = img.Height
6. numPixels = width * height
7. numBlocks = numPixels / blockPixeks
8. Set x and y to 1
9. for (i = 0; i < numBlocks; i++)

```
 currBlock = i % blockPixels
  if ( currBlock = = 0 ) // Key Block
    V = img.getPixel at position x and y
    keyStartPixel = no: of ones of binary
                    number of V
  Endif
  if (currBlock  ≠ 0) // Message Blocks
    Set x and y to new values
    V = img.getPixel at position x and y
    msgStartPixel = no: of ones of binary
                    number of V
  Endif
   x + = 3
   if (x ≥ width)
      x ← 1
      y + = 3
   Endif
 Endfor
Begin
 msgBit = GetMessageBits
 Begin
  Find the first message block
  lsbUsed = Get the last 4 bits of 8 neighbors
  for (i = 0; i < lsbUsed; i++)
    if (lsbBits [i] = msgBits [i])
      difference = 0;
    else
       difference += 1;
  Endfor
  Return difference
  Get the next message block and repeat
 End
Compare differences between 8 message
blocks
Mark message block with minimum
difference into first position of  key block
Get next message bits to find the next
minimum difference between the remaining
message blocks
```

End

2.4.2. Embedding Algorithm of OPMM

Bincvr = Binary number of pixel intensity value (V)
Input : Cover Image(C), Message (MSG).
Find the first seed pixel P_{rc}.
count = 1.
while (count ≤ n)
begin (for embedding message in message surrounding a seed pixel).
m_k = Get first msg bit.
count = count + 1.
Mask the 6^{TH} bit from left with the m_k in 'Bincvr'
m_{k+1} = Get next msg bit.
count = count + 1.
Mask the 5^{TH} bit from left with the m_{k+1} in 'Bincvr'
m_{k+2} = Get next msg bit.
count = count + 1.
Mask the 8^{TH} bit from left with the m_{k+2} in 'Bincvr'
m_{k+3} = Get next msg bit.
count = count + 1.
Mask the 7^{TH} bit from left with the m_{k+3} in 'Bincvr'
End
Get the next neighbor pixel $P_{r'c'}$ for embedding based on previous $P_{r'c'}$ and repeat.
End
Return the stego image (S).

2.4.3. Extraction Algorithm of OPMM

Bincvr = Binary number of pixel intensity value (V)
Input : Stego image (S) , count.
count = count ÷ 2.
BinMsg = " ".
Find the first seed pixel P_{rc}.
I = 0.
while (count ≤ N)
begin (for extract message in message around a seed pixel).
Get the (First/Next) neighbor pixel $P_{r'c'}$.
Bincvr = Binary of V.
Binmsg(i) = 1^{st} Bit of Bincvr from Right.
i = i+1.
Binmsg(i) = ZerothBit of Bincvr.
i = i+1.
Binmsg(i) =3^{rd} Bit of Bincvr from Right.
i = i + 1.
Binmsg(i) = 2^{nd} Bit of Bincvr from Right.
i = i + 1.
count = count + 1.
End.
Get the next neighbor pixel $P_{r'c'}$ for extracting based on previous $P_{r'c'}$ and repeat.
End loop.
Binmsg is converted back to Original message.
Return Original Message.
End.

III. RESULTS AND DISCUSSION

This system is implemented as a series of interfaces to show the data embedding approach for message authentication system. Some of them are illustrated in this section.

There are two processes for embedding messages into cover image. The first one is hash value generating on the concatenation of original message (M) and secret value (s). Then, the next is OPMM based embedding process.

Before embedding process using OPMM, the sender and receiver must share secret value (S). Firstly, the sender needs to type the value of S. Secondly, The original message and S are concatenated to generate the hash code by using SHA-512 hash algorithm. This process is shown in **Fig.6**.

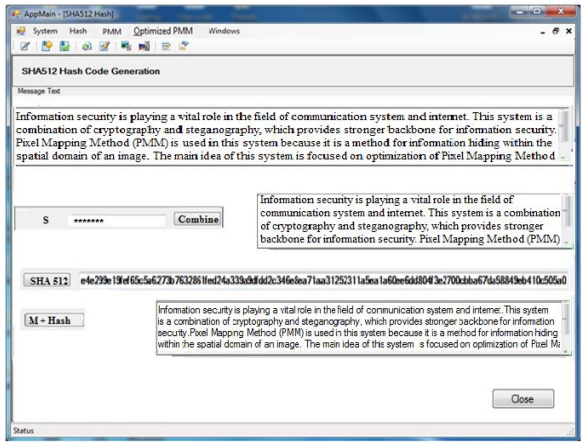

Fig.6. Combining Hash Value and Message

After that, the sender must load the cover image and the concatenated value (M || H (M || S)) are hidden into cover image through OPMM embedding algorithm to obtain the stego-image as shown in **Fig.7**.

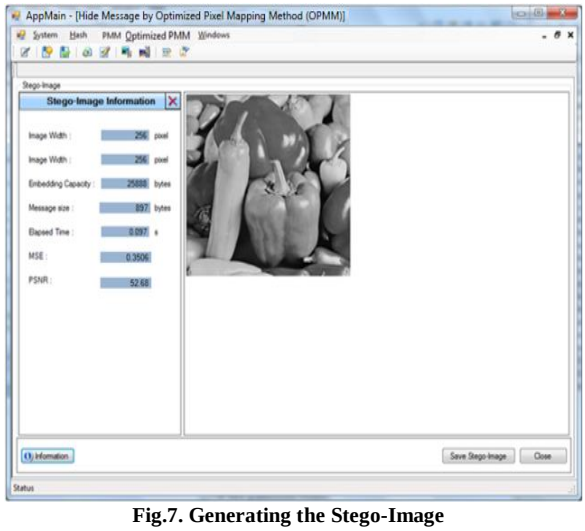

Fig.7. Generating the Stego-Image

In extraction process, the receiver needs to load the stego-image to retrieve the concatenated value from it using OPMM extraction algorithm.

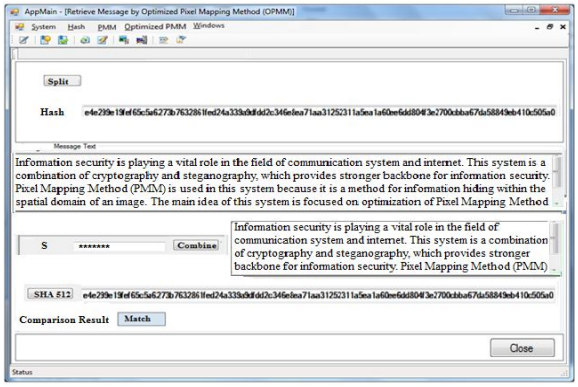

Fig.8. Checking the Data Integrity

Then, the extracted hash code H (M || S) is compared with the calculated hash code H' (M || S), which is obtained from the hashing of extracted message M and secret value S. If the two hash codes are matched, it is ensured the data integrity as shown in **Fig.8**.

In this section, performance consideration is made by two portions: embedding capacity consideration and the calculation of PSNR values for stego-image.

(1) Consideration on Embedding Capacity

The embedding capacity consideration depends on the following parameters:

Total number of pixels (P_t)

Block Size (B_s) = seed Pixel + eight neighbours pixels

Total number of blocks (B_t) = P_t / B_s

Number of unused blocks (B_{unused}) = B_t mod B_s

Number of key blocks (B_k) = P_t / B_s^2

Eight neighbors surrounding the seed pixels (n)

Number of hidden bits in each pixel (t)

The embedding capacity can be calculated by using the following formula.

Embedding Capacity = [B_t – B_{unused} – B_k] × n × t

To do the comparative study of OPMM and PMM for embedding capacity, the experiment is done in three different sizes of pepper image. According to the experimental results, it can be clearly seen that OPMM provides better embedding capacity as shown in **Fig.9**.

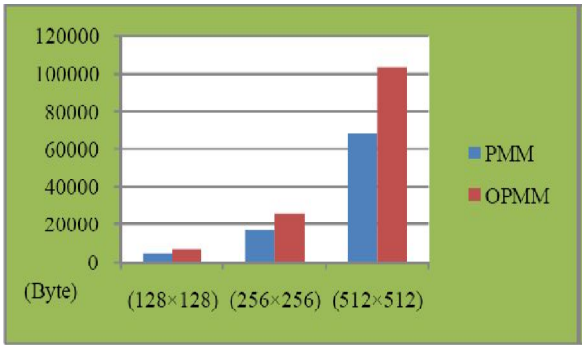

Fig.9. Comparative Study of Embedding Capacity

(2) Peak Signal to Noise Ratio (PSNR)

The PSNR is used to evaluate the quality of the stego image after embedding the secret message in the cover. It is assumed a cover image C (i, j) that contains N by N pixels and a stego-image S (i,j) where S is generated by embedding/mapping the message bit stream. Mean squared error (MSE) of the stego-image is calculated as follow:

$$MSE = \frac{1}{[N \times N]2} \sum_{i=1}^{N} \sum_{j=1}^{N} [C(i,j) - S(i,j)]^2$$

The PSNR is computed as follow:

$$PSNR = 10 \log_{10} 255^2 / MSE \text{ db}$$

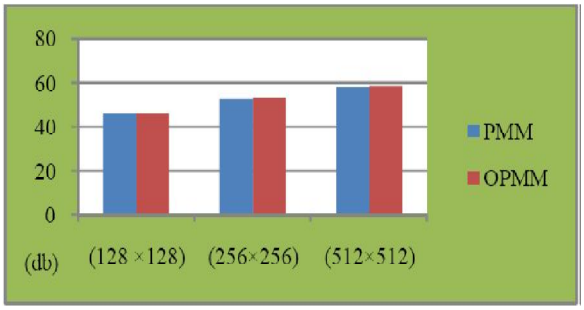

Fig. 10 Comparative Study of PSNR Values

The comparative results of PSNR are depicted in **Fig.10**. According to the experimental results, the PSNR values of OPMM are slightly higher than that of PMM. So, the OPMM also provides secret message embedding without producing major changes of stego-image quality like PMM.

CONCLUSIONS

In developing the proposed system, SHA-512 hash algorithm and OPMM embedding algorithm are integrated to meet the security requirements: confidentiality and message authentication. Moreover, the proposed system provides larger embedding capacity with minimum degradation of stego-image quality by using proposed embedding approach (OPMM). However, the proposed embedding method (OPMM) also processes only gray scale image like PMM. As further extension, it can be enhanced to process in the color images. The proposed system can be effectively applied in various security awareness areas such as commercial, medical imaging and military communication systems and so on.

REFERENCES

[1] William Strallings, “Cryptography and Network Security, Chapter 11,Cryptographic Hash Functions”, Fourth Edition. Richard Spillman, “Classical and Contemporary Cryptography”, Prentice Hall, 2005.

[2] Pragya Agarwal, Shilpi Gupta and Anu Mehra, “Transmission and Authentication of Text Messages through Image Steganography,” IJCA Proceedings on 4th International IT Summit Confluence, No.2, pp.16-20, 2013.

[3] Kran Bailey Kevin Curran, “An evaluation of image based steganography methods,” International Journal of Digital Evidence, Fall 2003, 2003.

[4] Jr. L. M. Marvel, C. G. Boncelet and C. T. Retter, "Spread spectrum image steganography," *IEEE Trans. on Image Processing*, 8:1075–1083, 1999.

[5] Nasir Memon R. Chandramouli, "Analysis of lsb based image steganography techniques," In Proceedings of IEEE ICIP, 2001.

[6] Wang, X.; Yin, Y.; and Yu, H. "Finding Collisions in the Full SHA-1". *Proceedings, Crypto '05*; published by Springer-Verlag, 2005.

[7] Souvik Bhattacharyya and Gautam Sanyal, "A data hiding model with high security features combining finite state machines and PMM method," *International Journal*, vol. 4, pp. 324–331, 2010.

★★★