A Proposed Encoding and Hiding Text in an Image by using Fractal Image Compression

Dr. Fadhil Salman Abed
Lecturer in University
Technical Institute of Kalar
Iraq-Diyala-Jalawla
E-Mail: Fad_Sal_Abed@Yahoo.com

Keywords
Image processing, Hiding, Fractal Image Compression, Quadtree, Steganography

Abstract
The problem of protecting data and information originates from the dependency upon encryption and steganography separately. Therefore, it is easy to break them and detecting the contents of the secret messages available in data, pictures, texts or audios.

To add multiple layers of security our research proposed an algorithm that can hide a text in an image in a such away that prevents as much as possible any suspicion having a hidden text, after RSA cipher.

In this research a new hiding method is proposed based on encoding the information by using RSA cryptosystem and hiding the cipher information(ciphertext) by using proposed fractal image compression(FIC). FIC based on IFS exploits the spatial redundancy with it is different parts and they are blocked based algorithm with variable block size that intend to approximate blocks of a determined size with contractive transformation. This blocks based allow hiding capacity to be large.

1.0 Introduction
With the increases in the number of digital networks and recording devices, digital images appear to be a material for steganography and information hiding techniques have been developed to be a strong basis for steganography area.

The aim of this paper is to describe a method for integrating together cryptography and Steganography for secure communication using an image file. The proposed system first compresses the secret message (i.e. word document) and then implements cryptographic algorithms to the compressed message. The resulted file is used as the secret message to be hidden in the digital an image file[1].

This research project produces new information hiding scheme and it consists of three parts, the first part is fractal Image Compression(FIC) which needs several stages of operations to be done on the colour image to find out the best approximate the affine transform is used to determine the necessary scale coefficient, offset coefficient and the isometric mapping needed to achieve best match between the range and domain blocks then the process of embedding the information have to be done.

The second part is implements cryptographic algorithms by using public-Key type RSA cryptosystem to the compressed message[2].

The third part is embedding cipher information(Plaintext) in an image. A new scheme has been produced for hiding by using the scaling and offset values of the blocks as a cover for hiding by getting secret text (character by character) and converting each character into its ASCII representation, then store them in to data array and adding the characters to the fraction part of scaling value without affecting the quality of the image. It can achieve a very high hiding rate of information.

In order to obtain the best approximate for each non-overlapped block of range, all overlapped blocks of the domain should be tested, taking into consideration that there are eight isometric possible states for each cover block and all of these possible states should be tested when trying to find the closest approximate. In other words good approximate can be obtained when many domain blocks are involved in the approximation process taking into consideration that the process of searching the pool of domain time consuming process.
2. Combination of cryptography and Steganography for secure communication

Combination of cryptography type RSA cryptosystem and Steganography by using FIC for secure communication is an application, which combines both Cryptography methods (i.e. Encryption, Decryption) and Steganography techniques to make the communication more secure. The outcome of this paper is to create a cross-platform tool that can effectively hide a message (i.e. Word document) inside a digital image file. It is concerned with embedding information in a secure and robust manner. In this paragraph, we illustrate the cryptography and FIC (fractal Image Compression) for compress and hiding the information[1],[3].

2.1 Cryptography Stage

There are several types of asymmetric algorithms used in the computing world today. They may have different internal mechanisms and methods, but the one thing they do have in common is that they are all asymmetric. This means that a key is used to encrypt a message different from the key that is used to decrypt a message. RSA is a public key algorithm that is the most understood, easiest to implement, and the most popular when it comes to asymmetric algorithm.

2.1.1 RSA Algorithm

RSA involves a public and private key. The public key can be known to everyone and it is used for encrypting messages.

The secret message in this work ends by (#) to determine the end of the message in an extracting stage. Each letter in the message is converted to a decimal number by ASCII code, and encrypted by using the public key.

The keys for the RSA algorithm are generated in the following way[4],[5],[6]:

- Choose two distinct large random prime numbers $P$ and $q$.
- Compute $n = pq$, $n$ is used as the modulus for both the public and private keys.
- Compute the: $\phi(n) = (p-1)(q-1)$.
- Choose an integer $e$ such that $1 < e < \phi(n)$, and $e$ and $\phi(n)$ share no factors other than 1 (i.e. $e$ and $\phi(n)$ are prime), $e$ is released as the public key exponent.
- Compute $d$ to satisfy the congruence relation $de \equiv 1 \pmod{\phi(n)}$, $d$ is kept as the private key. The public key consists of the modulus $n$ and the public (or encryption) exponent $e$. The private key consists of the modulus $n$ and the private (or decryption) exponent $d$ which must be kept secret. All parts of the private key must be kept secret in this form. The decoder can recover $m$ from $c$ by using her private key exponent $d$ by the following computation:

$$m \equiv c^d \pmod{n}.$$ Given $m$, the decoder can recover the original message $m$. 

Figures (1): Combination of cryptography and Steganography
This shows that we get the original message back:

\[ c^d \equiv m \pmod{n} \]

The public key is the product of two randomly selected large prime numbers, and the secret key is the two primes themselves. The algorithm encrypts data using the product, and decrypts it with the two primes, and vice versa. A mathematical description of the encryption and decryption expressions is shown below:

Encryption: \[ C = m^e \pmod{n} \], Decryption: \[ m = C^d \pmod{n} \]

Where:

- \( m \): The plain-text message,
- \( c \): the encrypted message expressed as an integer number.
- \( n \): the product of two randomly selected, large primes \( p \) and \( q \).
- \( e \): a large, random integer relatively prime to \( (p-1)*(q-1) \).
- \( d \): the multiplicative inverse of \( e \), that is:
  \[ (e * d) = 1 \pmod{(p-1)*(q-1)} \].

The public key is the pair of numbers \( (n,e) \). The private key is the pair of numbers \( (n,d) \).

### 2.2 Fractal Image Compression Stage

In fractal image compression the image to be coded is partitioned into blocks called ranges. Each range is approximated by another part of the image called domain. Finding a partitioning that minimizes the approximation error while not exceeding a given bit rate is a hard problem in FIC.

The goal of utilizing image partitioning in FIC is to approximate the pixel intensities of the range block with those of the domain block. Good approximation can be obtained when many domain blocks are involved in the matching (approximation) process. Taking into consideration that the process of searching the pool of domain blocks is time consuming (greedy) process, [7].

#### 2.2.1 Range Pool Generation

This module consists of two sub-modules, image loading and range partitioning.

#### 2.2.2 Image Loading

At first the image to be coded must be loaded as a two dimensional array. After loading the array, the image data stored in this two-dimensional array will be converted to a gray level image. This array is called the "Range".

#### 2.2.3 Range Partitioning (Range Pool)

Before the image to be coded using PIFS (Partition Iterated Function System), it must be partitional into non-overlapping block of different sizes, by using one of the adaptive partitioning methods. The resulting blocks are called "Range Blocks" (non-overlapped blocks).

In the quadtree fractal compression method, the partitioning depends on the maximum and minimum allowable block size. As a special case, quadtree partitioning becomes fixed partitioning; when the adopted maximum and minimum allowable block sizes are same.

#### 2.2.4 Domain Pool Generation

This module is responsible for generating another two-dimensional array \((Hd \times Wd)\), but with quarter the size of the range (i.e., \(Hd=H/2\) and \(Wd=W/2\)). This array is called the “Domain”.

#### 2.2.5 The Search

For each range block \(R\) the optimal affine approximation (mapping) can be represented as [8]:

\[ R_i \approx sD_i + o \]

Where \( R_i \) is the range pixel value,

- \( D_i \) is the corresponding domain pixel value,
- \( s \) is the scaling coefficient,
- \( o \) is the offset coefficient.

For mapping each range block, the search process implies that for all domain blocks \((D_i)\), listed in the domain pool, should be matched with the considered block, by computing the optimal approximation \((R_i \approx sD_i + o)\).
o). The eight symmetric versions of the domain block should be taken into consideration. Using the following
isometric mappings produces the symmetric blocks:

1. Identity: \( \text{Sym}(x, y) = R(x, y) \)
2. Rotation (+90): \( \text{Sym}(x, y) = R(y, BS-x) \)
3. Rotation (+180): \( \text{Sym}(x, y) = R(BS-x, BS-y) \)
4. Rotation (+270): \( \text{Sym}(x, y) = R(BS-y, x) \)
5. Reflection about mid-vertical axis: \( \text{Sym}(x, y) = R(y, x) \)
6. Reflection and rotation (-90): \( \text{Sym}(x, y) = R(x, BS-y) \)
7. Reflection and rotation (-180): \( \text{Sym}(x, y) = R(BS-y, BS-x) \)
8. Reflection and rotation (-270): \( \text{Sym}(x, y) = R(BS-x, y) \)

where \( BS \) is the size of the domain block, \( \text{Sym} \) is the symmetric of block.

The optimal approximation could be done as follows:

1. Compute the scale (s) and offset (o) coefficients, using the following equations:

\[
\begin{align*}
\frac{1}{n^2} \left( \sum_{i=1}^{n} d_i r_i \right) - \left( \sum_{i=1}^{n} d_i \right) \\
\left( \sum_{i=1}^{n} r_i \right) - \left( \sum_{i=1}^{n} d_i \right)^2 \\
\end{align*}
\]

\[
\begin{align*}
s &= \frac{1}{n^2} \left( \sum_{i=1}^{n} d_i r_i \right) - \left( \sum_{i=1}^{n} d_i \right) \\
o &= \frac{1}{n^2} \left( \sum_{i=1}^{n} r_i \right) - s \left( \sum_{i=1}^{n} d_i \right)
\end{align*}
\]

2. Check the coefficients:
   a. for scaling coefficient:
      i. if \( s < s_{\text{min}} \) then \( s = s_{\text{min}} \)
      ii. if \( s > s_{\text{max}} \) then \( s = s_{\text{max}} \)
   b. for offset coefficient:
      i. if \( o < o_{\text{min}} \) then \( o = o_{\text{min}} \)
      ii. if \( o > o_{\text{max}} \) then \( o = o_{\text{max}} \)

3. Quantization

\[
\begin{align*}
s_q &= \text{round} \left( \frac{2^{b_s} - 1}{s_{\text{max}} - s_{\text{min}}} (s - s_{\text{min}}) \right) \\
o_q &= \text{round} \left( \frac{2^{b_o} - 1}{o_{\text{max}} - o_{\text{min}}} (o - o_{\text{min}}) \right)
\end{align*}
\]

where:

- \( b_s \): number of bits assigned to scaling coefficients, \( b_o \): number of bits assigned to offset coefficients,
- \( s_{\text{max}} \): maximum value of scaling coefficients, \( s_{\text{min}} \): minimum value of scaling coefficients,
- \( o_{\text{max}} \): maximum value of offset coefficients, \( o_{\text{min}} \): minimum value of offset coefficients.

4. Compute the approximation error (\( d_{\text{RMS}} \)) using equation:

\[
\begin{align*}
E^2(R, D) &= \frac{1}{n^2} \left[ \sum_{i=1}^{n} r_i^2 + s \left( \sum_{i=1}^{n} d_i^2 - 2 \sum_{i=1}^{n} r_i d_i + 2 \sum_{i=1}^{n} d_i \right) + o \left( o^2 - 2 \sum_{i=1}^{n} r_i \right) \right]
\end{align*}
\]

5. Compare the computed error with the minimum registered error (\( d_{\text{min}} \)) i.e., if \( d_{\text{RMS}} > d_{\text{min}} \) then jump to step (7).

6. Register

\[
\begin{align*}
s_{\text{opt}} &= s_q, \quad o_{\text{opt}} = o_q, \quad d_{\text{RMS}} = d_{\text{RMS}}, \quad S_m = \text{Symmetry index,}
\end{align*}
\]

\[
\begin{align*}
x_d &= x \ (\text{the coordinate of the current tested domain block}), \\
y_d &= y \ (\text{the coordinate of the current tested domain block}).
\end{align*}
\]

7. Repeat the steps (1) to (5) for all symmetry versions of the tested domain blocks.

8. Repeat the steps (1) to (7) for all domain blocks listed in the domain pool.
9. Output the set of parameters \((s_{opt}, o_{opt}, x_d, y_d, \text{ and } S_m)\) as a set of fractal coding parameters for the tested range block

3.0 The Proposed Hiding and Encoding System

Figure (2) describes the proposed system starting with Encoding Text by Using RSA Cryptography and loading cover image, then performing the quad-tree partition and then hiding module by starting search process for similarity between the image blocks (range blocks and domain blocks) and embedding the secret message in the scaling and offset values of the blocks. The output of this stage is a data file of stego-image which is sent from a sender to a recipient. When the recipient receives the data file of the stego-image, the process of extraction could be applied to obtain the secret message with an approximate or the reconstructed image.

![Figure (2): Embedding Information Unit](image)

3.1 Secret Message Embedding

For embedding the secret message, a new method has proposed and in this new method a huge number of characters can be embedded, beside the characters or the secret message there may not necessary be a text, it may consist of equation or numbers with another language[9]. After the IFS mapping is coming to the end of the last block and the parameter values have been set, the process of embedding starts by reading the inputted secret message and converting it to its binary representation then store them in a new array separately.

Next, the secret message (SecData) characters are taken one by one and they are converted to its ASCII representation. The length of the secret message is limited to the number of blocks. As the number of blocks increase in tern more characters can be embed, in other words this embedding method depends upon the number of blocks.

\[
\text{Length of secret message (SecData) } = (\text{No. of blocks } \times 2) - 1
\]

Each character is embedded in the scaling value of the blocks by taking the integer part of the scaling using this equation.

The Proposed Hiding and Encoding System include the following steps:-
1. Load a colour image (bitmap format 24 bits), and part the colours into the red, green, and blue.
2. Convert the image formula from RGB to YCbCr[10]
   \[
   Y = (77/256) R + (150/256) G + (29/256) B
   \]
Cb = -(44/256) R – (87/256) G + (131/256) B + 128
Cr = (131/256) R – (110/256) G – (21/256) B +128

3.  Partitioning the cover image  by using  Quad-tree Partition the algorithm illustrated
in figure (8) in appendix.

  Then taking the two digit of the fraction part (2 digit after the decimal point) of the value and
neglecting the other digits
  5. $\text{IntFraction} = \text{fix}(V \times 100)$
  6. $\text{Fraction} = \text{IntFraction} \times 0.01$
  Now the secret message will be added to the fraction value in order to occupy the 3rd, 4th and 5th
places after the decimal point without affecting the 1st and 2nd places of the original fraction.
  7. Encoding(Cryptography) the input message by using RSA cryptosystem, the output is called
$\text{EncSecData}(I)$.

8. Calculate Iterated Function System(IFS)
   $\text{IFS}(I). Scl = \text{Fix}(\text{IFS}(I).Scl) + \text{Fraction} + \text{EncSecData}(I) \times 0.00001$
   The output is a set of scale and offset values that contain the secret massage (SecData) is the last
stage in hiding unit.

3.3 Information Extraction
   After the  data file (stego-image) has been loaded, the process of reconstructing SecData is applied to
extract the array of embedded secret characters, which have been stored in the (IFS) coefficients (s, o) in a
reverse way. This stage implies the following steps:

1. Extract the two digit of the fraction part of the coefficient (s and o) with keeping the integer part.
   $V = IFS(I).scl – \text{Fix}(IFS(I).scl)$
   $V_s = v \times 1oo$
2. Decoding the secret massage by using RSA decoding Algorithm

3. Convert the extracted data to byte
   $\text{DecSecData}(I) = \text{CByte}((V_s – \text{Fix}(V_s)) \times 1000$
4. Convert the bytes to string representation
   $\text{recSecData} = \text{recSecData} & \text{CStr(Chr(DecSecData(I))})$
5. Display the secret message.

3.4 Saving Reconstructed Image
   When the completion of all the above decoding steps, the secret message reconstructed as an output
beside the reconstructed cover image.

3.5 System Implementation
   The goal of this system is to Encoding Text by Using RSA Cryptography and hiding information (text,
numbers, symbols or equations) in a cover-image (BMP format) after compressing the image to produce the
stego-image as a data file.
   System implementation accepts seven inputs in the embedding stage:
1. Loading the cover image (BMP.format) as the input file.
2. Input control parameters.
3. Quad-tree partitioning the colour component.
4. Domain generating.
5. Inputting the secret message for embedding.
6. Encoding secret message to be embedding by using RSA cryptosystem.
7. Fractal encoding which include embedding.

System implementation accepts one input in the extracting stage:
1. Input the data file which contains the secret message beside image data array.
2. Extracting the secret message and reconstructing the image in the same time.

3.6 System Requirements

The Microsoft Window XP has been used as an operation system and Visual Basic (VB6) as a programming language.

3.7 System Steps

The proposed system starts operation with the general form interface of the project

- Partitioning the Cover Image

After the cover-image has been chosen, control parameter will be entered to perform quad-tree partition for each colour component (R component, G component, B component) separately, Flowchart (1) in Appendix illustrates the quadtree partitioning procedure.
- **Generating the Domain Image**

Generating the domain image and domain pool is next to the partition step, the domain size taken is quarter the image size with overlapped blocks.

![Figure (4): Domain Image Generating](image)

- **Input Secret Message**

Before starting fractal encoding, the message must be written in order to be embedded after the scale and offset values will be stored.

![Figure (5): Entering the Secret Message for Embedding](image)

- **Fractal Image Compression**

Searching for similarity is performed between the range and the domain blocks and the information is stored in an index, then the image (cover-image) information is stored as a structure array of data containing the secret message.

The received data is a collection of data that represent the image with the secret message. The receiver will extract the embedded information (secret message) and then reconstruct the cover image.
3.7 The Effect of Block Size

In these sets of tests it is focused on finding out how the quad-tree partitioning is affected when different values of control parameters are adopted, Table (1) shown that the number of blocks produced by quad-tree partition depends on the adopted partitioning parameters (maximum block size, minimum block size, inclusion factor and acceptance ratio). For hiding the secret message, the number of blocks determine the length of the secret message, for obtaining the large number of blocks small values must be chosen which mean more characters will be hidden (according to the number of blocks).

The increase of the number of blocks increase the encoding time and PSNR but decrease the RMSE also the compression ratio.
3.8 The Effect of Hiding Secret Message

The main objective of the proposed encoding and hiding scheme is to embed a secret message with a huge number of characters as possible with different languages and numbers without degrading the quality of the reconstructed cover image. So to evaluate the effect of the secret message embedding on the cover image, a set of tests is applied. Table (2) show the result of hiding different message process.

<table>
<thead>
<tr>
<th>No. char</th>
<th>Type</th>
<th>Time(sec)</th>
<th>RMSE</th>
<th>PSNR1(db)</th>
<th>PSNR2(db)</th>
<th>PSNR3(db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9653</td>
<td>Enlishtext</td>
<td>113.537037</td>
<td>7.1480366</td>
<td>31.05610</td>
<td>31.05606</td>
<td>31.06429</td>
</tr>
<tr>
<td>10283</td>
<td>Arabictext</td>
<td>113.537037</td>
<td>7.1481766</td>
<td>31.69444</td>
<td>31.05585</td>
<td>31.06429</td>
</tr>
<tr>
<td>11390</td>
<td>Mixed</td>
<td>114.69444</td>
<td>7.1404833</td>
<td>31.06484</td>
<td>31.06464</td>
<td>31.06424</td>
</tr>
</tbody>
</table>

4.0 Conclusions

During the implementation and testing phases of the proposed encoding and hiding system, the following remarks have been addressed:

1. Steganography is an effective way to obscure data and hide sensitive information. The effectiveness of Steganography is amplified by combining it with cryptography. By using the properties of the FIC Steganography algorithm for image file and combining it with the RSA cryptography standards, we developed a method, which adds layers of security to the communication. Steganographic methods do not intended to replace cryptography but supplement it.

2. The proposed system is build of muti-layer security, consequently even if the original copy of the stego-object is available, the intruder attacked by the second layer of security which is the encryption of the text embedded then he should analyze the cipher text using cipher text only attack. Which difficult to be analyzed.

3. The proposed system first compresses the secret message (i.e. word document) and then implements cryptographic algorithms to the compressed message. The resulted file is used as the secret message to be hidden in the digital image file.

4. The proposed system does not affect the image resolution; we can say it is not noticeable for human eyes. To prove this we show the cover-image and the stage –image to a team of 20 persons to take their opinion if
there is any different between the sego-image and the cover-image and their answer that there is no difference between both images.

References

Appendix

Input RGB Image

Transform to the desired color model

Y-component

Cb-component

Cr-component

Compute global mean and standard deviation

Set the partition control parameters; $S_{max}$, $S_{min}$, $I_c$, $R_u$

1

Flowchart (1): illustrates the quadtree partitioning procedure
Continuation of Flowchart (1)

1

Initialize LinkList record
LList (0). X=0; LList (0). Y=0
LList (0). Size=Width; LList (0). Next=-1;

Set I=0; NoList=1

Set X=LList (I). X; Y=LList (I). Y
S=LList (I). Size; J=LList (I). Next

I=LList (I). Next

If I = -1

Yes

No

Partitoning list for each color component

End

If S ≥ S_{min}

Yes

No

Compute local mean and determine no. of undesired points (N_p)

If N_p / S > R_p

Yes

No

Create new four LinkList records
S=S/2
LList (I). X=X; LList (I). Y=Y
LList (I). Size=S; LList (I). Next=NoList
LList (NoList), X=X+S; LList (NoList), Y=Y
LList (NoList). Size=S; LList (NoList). Next=NoList+1
LList (NoList+1). X=X; LList (NoList+1), Y=Y+S
LList (NoList+1). Size=S; LList (NoList+1). Next=NoList+2
LList (NoList+2). X=X+S; LList (NoList+2). Y=Y+S
LList (NoList+2). Size=S; LList (NoList+2). Next=J