The Fourth International Workshop on Image Media Quality and its Applications, IMQA2011
October 4-5, 2011, Kyoto, Japan

A STEGANOGRAPHIC METHOD USING MRF-SYNTHESIZED TEXTURES AS COVER IMAGES

Feng-Ju Chang and Chaur-Chin Chen

Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu 30013, Taiwan

Abstract

We adopt a grayscale statistical texture synthesizer based on MRF to generate an image of user-requested size to fit the secret message. Each of our synthesized texture images as cover images contains four gray values: 30, 100, 170, and 240. We encrypt secret messages via exponent and modulo operations. Then we partition the encrypted bit sequence of the secret message as many of 2-bit words: 00, 01, 10 and 11, thus each 2-bit word naturally corresponds to one of the four gray values 30, 100, 170 and 240. To be more secure, we adopt a circular shift technique on (30, 100, 170, 240) such that the same 2-bit words need not be embedded into the same pixel values and the security could be further ensured. Experiments are given to demonstrate our approach.

Keywords: Markov Random Field (MRF), Steganography, Texture

1 Introduction

Steganography [12] is a popular topic for scholars since Internet becomes the most common way of communication. It forces people to establish a passive attitude of protecting data with high security to avoid being victims. The most important requirement of steganography is undetectability; the concealed messages should be perfectly disguised under all statistical and visual analysis [5, 8, 11].

Since we discuss steganography on images, if images are always acquired from Internet, there is a problem that the length of message would be limited by the image size [1, 6]. To overcome this problem, a grayscale statistical texture synthesizer based on Markov random field (MRF) [1, 2, 3] is adopted such that we can synthesize the designated size of image needed.

Reversible data hiding [7] which ensures that after extracting the messages the image could preserve its primordial pattern. It takes the pixels with peak value in the histogram as targets for embedding. This paper uses a cover image of user-selected size and we want to develop a steganographic method based on the ideas of reversible data hiding algorithm and quantum cryptography [10] that every quantum would be affected by other quanta with a unique relationship respectively to increase the security. For the encryption method, we shall adopt the exponent and modulus computations [4, 9]. Figure 1 depicts our scheme.

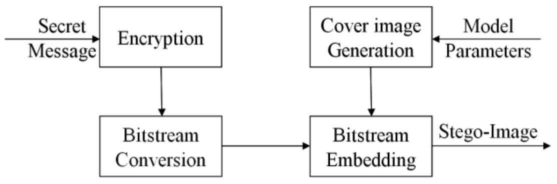


Figure 1. A paradigm of data hiding scheme.

2 Background Review

2.1 Texture Synthesis Based on Markov Random Field

Markov Random Field [2, 3] is a Gibbs random field which represents a dependent relationship of the pixels within the defined neighborhood. Taking advantage of properties of MRF to synthesize a meaningful texture, we define the renewed probability with its energy function U(x) introduced below; it would make the pattern related to parameters we give. Before showing the energy function U(x), some notations are listed as follows

$A = \{0,1,2,\dots,G-1\}$, a set of gray values

x: an initial $M \times N$ texture in matrix form and $x(i,j)$ belongs to A

$\Omega = \{x \mid x_t = x(i,j) \in A\}$, the set of all possible occurrences

c: the size of neighborhood, here $c = 2$ means the 2nd − order neighbordood

θ_r: the weight of direction r

$F(x_t) = \alpha_{x_t}$: the weight of a pixel value x_t

$r = \min\{1, P(y)/P(x)\}$, the renewed probability

The probability of each $x \in \Omega$ can be written as

$$P(x)=\frac{e^{-U(x)}}{Z}, \text{ where } Z=\sum_{y\in\Omega} e^{-U(y)}$$

$$U(x) = \sum_{t=1}^{MN} F(x_t) + \sum_{t=1}^{MN}\sum_{r=-c}^{c} H(x_t,x_{t:+r})$$

$$H(a, b) = H(b, a), H(x_t,x_{t:+r})=\theta_r I(x_t,x_{t:+r})$$

$$I(a, b)=\begin{cases} -1 & \text{if } a = b, \\ 1 & \text{otherwise}' \end{cases}$$

$$P(x_t|R_t) = \frac{\exp[-\alpha_{x_t} - \sum_{r=-c}^{c} \theta_r I(x_t,x_{t:+r})]}{\sum_{s\in A} \exp[-\alpha_s - \sum_{r=-c}^{c} \theta_r I(s,x_{t:+r})]}$$

Algorithm of texture synthesis based on MRF [2]

```
(1) For s = 1: MN, randomly assign a $g \in A$ for $x_s$ to give an initial image x,
(2) For s = 1: MN, Do
    (a) Let $y_t = x_t$ for all $t \neq s$. Choose $g \in A$ at random and let $y_s = g$,
    (b) Let $r = \min \{ 1, P(y)/P(x) \}$,
    (c) $x \leftarrow y$ with probability r.
(3) Repeat step (2) until "convergence" or K iterations, for example, K = 50.
```

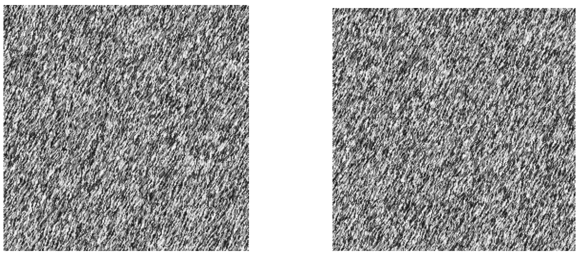


Figure 2. MRF-synthesized textures with the parameters (-1, 1, 0, 1).

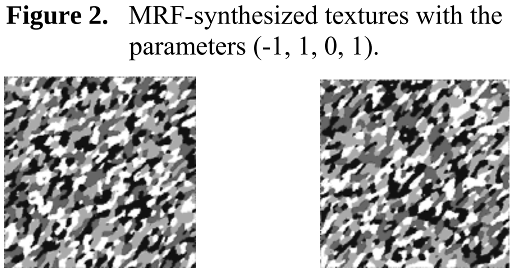


Figure 3. MRF-synthesized textures with the parameters (1, 1, 0, 1).

2.2 Discrete Logarithm Problems

In encryption, it's very easy to compute the modular exponentiation operations via Eq. (1). But in decryption, the inverse operation of encryption would force hackers to face an arduous task. That is, with a large prime p, even the hacker knows y, g and p, without the additional information, it's almost an impossible mission for finding x.

$$y \equiv g^x \bmod p, \text{ where} \quad (1)$$

x: an integer to represent the original message

y: the substitution of an encrypted message

g: a primitive root

p: a large prime number

$\Phi(p)$: Euler Totient function which is defined as the number of positive integers less than p and relatively prime to p.

In Eq. (1), there exist Φ (p-1) primitive roots [9] which possess the following property.

If p is a prime and g is a primitive root of p with
$$2 \le g \le p - 1, \text{then}$$
$$\{g^x \bmod p, \text{for } 1 \le x \le p - 1\} = \{1,2, \dots, p - 1\}.$$

With the property, if we take each character as an unsigned integer under the prime p = 257, we can encrypt messages into a sequence of meaningless codes successfully.

2.3 Reversible Data Hiding

The most important advantage of reversible data hiding is that after extracting messages the cover image can be completely recovered. In Ni's reversible data hiding [7], peak and valley are defined as the pixel values with the highest and lowest frequency in the image histogram, respectively. In the beginning, shifting all pixel values in (peak, valley) for one level to the right (or all pixel values in (valley, peak) for one level to the left), in embedding, it then scans bit stream to embed the bit one into pixel value peak+1 according to the positions of pixel visits, and skip the action of bit 0. This idea comes from seeing the peak values as targets for embedding, thus the capacity could be counted as the quantity of peak frequency.

2.4 Some Concepts of Quantum Cryptography

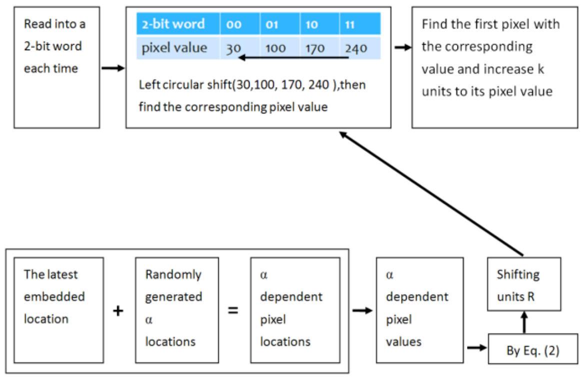
Quantum cryptography [7] is a secure protocol system utilizing the two quantum mechanical principles. First, the quantum state would be spoiled if anyone attempts to survey it. Second, in nature there exists a unique effect in any two quanta no matter how long the distance is. We adopt the concept of the second property to change the embedding target every time. In other words, the embedded target will be affected by some pixels randomly chosen each time to increase the security level.

3 Proposed Data Hiding Method

In the beginning, we set a natural correspondence of four pixel gray values in the order of (30, 100, 170, 240) with the four 2-bit words (00, 01, 10, 11) as shown in Table 1.

Table 1. A correspondence of natural order.

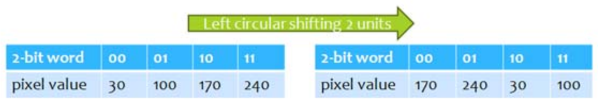
2-bit word	00	01	10	11
pixel value	30	100	170	240



Read into a 2-bit word each time
2-bit word 00 01 10 11
pixel value 30 100 170 240
Left circular shift(30,100, 170, 240),then find the corresponding pixel value
Find the first pixel with the corresponding value and increase k units to its pixel value
The latest embedded location
+
Randomly generated α locations
=
α dependent pixel locations
α dependent pixel values
Shifting units R
By Eq. (2)

Figure 4. A paradigm of embedding.

For each 2-bit word embedding, we left circular shift R units of the natural order (30, 100, 170, 240) such that the same 2-bit words need not be embedded into the same pixel values. An example of left circular shifting R = 2 units is illustrated in Figure 5.



Left circular shifting 2 units
2-bit word 00 01 10 11
pixel value 30 100 170 240
2-bit word 00 01 10 11
pixel value 170 240 30 100

Figure 5. An example of left circular shifting R=2 units.

The left circular shifting units, R, is determined by the latest embedded location. First, we randomly generate α locations which serve as a set of fixed relative locations. Second, use these α relative locations to get the dependent pixel locations of the latest embedding pixel location as demonstrated in Table 2. Then we substitute the pixel values of the dependent pixels into Eq. (2) to compute the left circular shifting units R; meanwhile, we can get the shifted correspondence between 2-bit words and four gray values. Because the pixel values corresponding to the 2-bit words vary along with the time, the security could be further ensured.

Table 2. Illustration of dependent pixel locations for an M × N cover image.

	The latest embedded location	+	α locations		=	Dependent pixel locations	
x	a		x_1	…		$a + x_1 \pmod M$	…
y	b		y_1	…		$b + y_1 \pmod N$	…

$$R = \sum_{i=1}^{\alpha} \lfloor (P_i - 30)/70 \rfloor, \text{where} \qquad (2)$$

P_i: the ith dependent pixel value,

α: the number of dependent pixels, α = 10 is used.

Embedding Algorithm

Input: a secret key K to generate α locations, and
Image (0: M-1, 0: N-1), an M×N MRF texture,
S: an (encrypted) secret message recorded as a sequence of 2-bit words,
k: an integer in [1, 9] to signify the embedding,
R: the units of left circular shifting, R=0 initially.

(1) Randomly generate α locations using a secret key K, set the initial embedding location as (0, 0), and scan the cover image in a specific order, for example, a lexicographic order.
(2) Read into a 2-bit word m from the binary sequence of secret message S each time.
(3) Left circular shift the pixel ordered values (30, 100, 170, 240) R units which corresponds to the 2-bit word order (00, 01, 10, 11).
Find the shifted pixel value corresponding to m.
(4) Follow the latest embedding location to search the next pixel location with the shifted pixel value to be the latest embedding pixel location and increase k to its pixel value.
(5) Get α dependent pixel locations by Table 2 according to the latest embedding location and α locations and compute R by Eq. (2).
(6) Repeat steps (2~5) until the sequence of 2-bit words for the message S are embedded.

An Illustrated Example

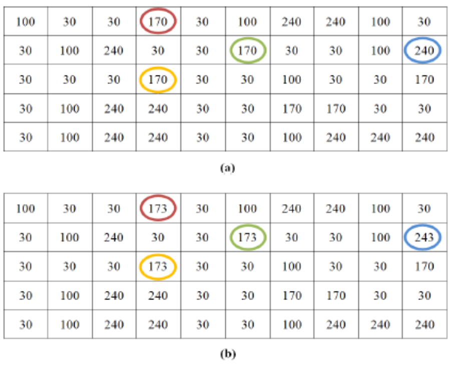
Image(0:4, 0:9) : a 5×10 cover image as shown in Figure 6(a), the top leftmost pixel location is recorded as (0, 0) with the pixel value 100.
S: Bit sequence of encrypted secret message {10, 11, 10, 01}.
α locations ={(3,0), (1,3)}, where α = 2.
k : is set to be 3 in this example.
Natural correspondence of 2-bit words and four gray values are as given in Table 1.
R: left circular shifting units determined by Eq. (2).

(1) To embed the first 2-bit word "10" with the initial R = 0, we find the corresponding pixel value 170 to match "10" from Table 1, we then scan the

cover image from (0,0) in the lexicographic order to find the first pixel value 170 which is in the location (0,3). We embed the "10" by updating Image(0,3)=173 from Image(0,3)=170. The next dependent locations relative to the current location (0,3) are (0,3) + (3,0) = (3,3) and (0,3) + (1,3) = (1,6) with the corresponding pixel values Image(3,3)=240, Image(1,6)=30, respectively. We then compute the next shift units R by Eq.(2) as $R = \left\lfloor \frac{240-30}{70} \right\rfloor + \left\lfloor \frac{30-30}{70} \right\rfloor = 3 + 0 = 3 \pmod 4$.

(2) To embed the next 2-bit word "11" with R = 3 computed in the previous step, we find the corresponding pixel value 170 which is corresponding to 2-bit word "11" obtained by left circularly shifting R=3 according to Table 1. Then scan the cover image from the latest embedded location (0, 3) in the lexicographic order to find the next location (1, 5) corresponding to the pixel value 170. We embed the "11" by updating Image(1,5)=173 from Image(1,5)=170. We then compute the next shift units R by Eq.(2) as R=0 (mod 4).

(3) Repeat the similar processes of (1) and (2), the contents of cover image and stego-image of this example are shown in Figure 6.



(a)

100	30	30	170	30	100	240	240	100	30
30	100	240	30	30	170	30	30	100	240
30	30	30	170	30	30	100	30	30	170
30	100	240	240	30	30	170	170	30	30
30	100	240	240	30	30	100	240	240	240

(b)

100	30	30	173	30	100	240	240	100	30
30	100	240	30	30	173	30	30	100	243
30	30	30	173	30	30	100	30	30	170
30	100	240	240	30	30	170	170	30	30
30	100	240	240	30	30	100	240	240	240

Figure 6. (a) a cover image, **(b)** the stego-image.

Extraction Algorithm

Input: a secret key K to generate α locations, and
Image (0: M-1, 0: N-1): an M×N stego-image,
k: an integer in [1, 9] used to signify the embedding,
R: the left circular shifting units for each 2-bit word used in embedding, and R=0 initially.

(1) Use the secret key K issued in embedding to generate α locations for computing dependent locations. Set the latest embedded (an initial location) as (0, 0).

Scan the stego-image in the same order as used in embedding.

(2) Follow the latest embedded location, find the next embedded pixel location (i, j) with pixel value g, g ∈ {30+k, 100+k, 170+k, 240+k }, decrease pixel value Image(i, j) by k.

(3) Left circularly shift the ordered pixel values (30, 100, 170, 240) R units corresponding to the 2-bit words (00, 01, 10, 11) according to Table 1. Find the 2-bit word corresponding to g-k.

(4) According to the latest embedded location and α locations to compute α dependent pixel locations by Table 2.

(5) Compute R by Eq. (3.1).

(6) Repeat steps of (2~5) till there is no image pixel ∈ {30+k, 100+k, 170+k, 240+k}.

Apply this extraction algorithm, the encrypted sequence consisiting of 2-bit words could be retrieved. Afterwards, the original message can be easily decrypted. The programs *hide* and *seek* are used for embedding and extraction, respectively.

4 Experimental Results

Since the four gray values contained in a cover image uniformly appear, for embedding each 2-bit word, we determine the minimum number of pixels of a cover image by Eq.(3) such that the probability of a successful embedding will be higher than 0.95.

$$\text{P(at least one corresponding pixel values)} > 0.95$$
$$1 - (0.75)^{L} > 0.95 \rightarrow L > 10 \quad (3)$$

To make the probability of a successful embedding higher than 0.95 every 2-bit word needs 11 pixels for embedding at least. If we have T characters in a secret message, a cover image of approximately 44*T pixels would be enough to embed the secret message. An article of 4574 characters, captured from http://www.cnn.com [13] which is also available as msg.txt [14], was used as secret message [14] for embedding into a 256x256 MRF synthesized texture, the cover image and the stego-image are shown in Figure 7. The corresponding histograms of image in Figures 7(a), (b) in Figure 8 demonstrates that the message is uniformly embedded into the pixel values.

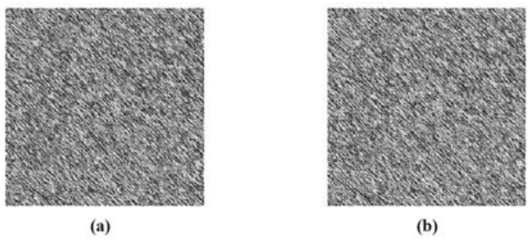


Figure 7. (a) a cover image, **(b)** a stego-image with parameter (0, 0, 1, -1) for MRF with the image size 256×256, k=3, the message contains 4574 characters.

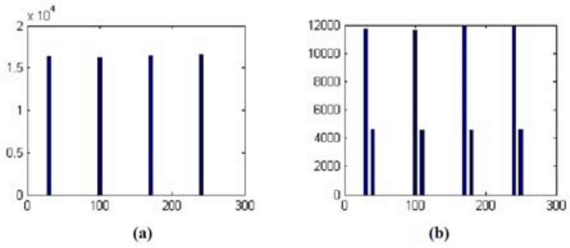
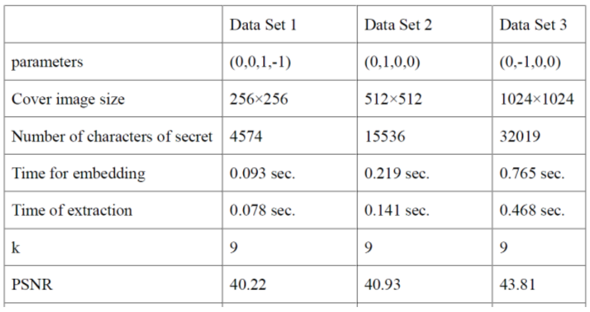


Figure 8. (a) histogram of Figure 7(a), **(b)** histogram of Figure 7(b).

The parameters used for synthesizing MRF textures, the size of three message data sets, the time for embedding and extraction associated with the PSNR value of measuring the difference between a cover image and its stego-image are summarized in Table 3.

Table 3. Summary of Experimental Results.



	Data Set 1	Data Set 2	Data Set 3
parameters	(0,0,1,-1)	(0,1,0,0)	(0,-1,0,0)
Cover image size	256×256	512×512	1024×1024
Number of characters of secret	4574	15536	32019
Time for embedding	0.093 sec.	0.219 sec.	0.765 sec.
Time of extraction	0.078 sec.	0.141 sec.	0.468 sec.
k	9	9	9
PSNR	40.22	40.93	43.81

5 Conclusion

This work addresses generating any user-requested size of texture image, based on Markov random field synthesis, as a cover image to meet the size of secret message. Furthermore, if there are some pixels unused in the later rows of a stego-image, we can cut them without changing the visualization of stego-images. Second, each texture synthesized with the same parameters looks visually the same but different in their contents. Third, the same 2-bit words need not be embedded into the same pixel values. Without knowing α locations, it's hard to extract the binary encrypted secret message sequence. In summary, if a cover image is also part of secret information, then our approach of using MRF-synthesized textures provides a solution.

6 Acknowledgments

This work is supported by research Grants NSC 97-2221-E-007-122-MY3 and NSC 100-2221-E-007-118. We thank the valuable suggestions from the reviewers.

7 References

[1] F. J. Chang, A Steganographic Method Using MRF-Synthesized Textures as Cover Images, M.S. Thesis, National Tsing Hua Unviersity, Hsinchu, Taiwan, April 2011.

[2] C.C. Chen and C.C. Chen, ‘’Texture Synthesis: A Review and Experiments,” Journal of Information Science and Engineering, Vol. 19, No. 2, 371 - 380, 2003.

[3] G.R. Cross and A.K. Jain, “Markov Random Field Texture Models,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 5, No. 1, 25 - 39, 1983.

[4] D.M. Gordon, “Discrete logarithms in GF(p) using the number field sieve,” SIAM J. Discrete Math, Vol. 6, No. 1, 124 - 139, 1993.

[5] N.F. Johnson and S. Jajodia, “Steganalysis of images Created Using Current Steganographic Software,” Int’l Workshop in Information Hiding, Berlin, Vol. 1525, 273 - 289, 1998.

[6] C.L. Liu and S.R. Liao, “High-performance JPEG steganography using complementary embedding strategy,” Pattern Recognition, Vol. 41, 2945-2955, 2008.

[7] Z. Ni, Y.Q. Shi, N. Ansari, and W. Su, “Reversible data hiding,” IEEE Transaction on Circuits and Systems for Video Technology, Vol. 16, No. 3, 354 - 362, 2006.

[8] N. Provos and P. Honeyman, “Hide and seek: an introduction to steganography,” IEEE Security & Privacy, Vol. 1, No. 3, 32 - 44, 2003.

[9] D.R. Stinson, “Cryptography: Theory and Practice, Champman&Hall/CRC Press, 2006.

[10] D. Stucki, N. Brunner, N. Gisin, V. Scarani, and H. Zbinden, “Fast and simple one-way quantum key distribution,” IEEE Applied Physics Letters, Vol. 87, No. 19, 194108-1 - 194108-3, 2009.

[11] A. Westfeld, “F5 – A Steganographic Algorithm: High Capacity Despite Better Staganalysis,” International Workshop on Information Hiding, Berlin, Vol. 2137, 289-302, 2001.

[12] http://en.wikipedia.org/wiki/Steganography, last access on September 6, 2011.

[13] http://edition.cnn.com in March 2011.

[14] http://www.cs.nthu.edu.tw/~cchen/msg.txt