http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (15): 2404-2410, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.2404.2410

An Automatically Changing Feature Method based on Chaotic Encryption

[1]Wang Li, [1]Gang Luo and [2]Lingyun Xiang
[1]College of Information Science and Engineering, Hunan University, Changsha, 410082, China
[2]College of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China

Abstract: In practical applications, in order to extract data from the stego, some data hiding encryption methods need to identify themselves. When performing data hiding, they embed some specific logo for self-identification. However, it is unavoidable to bring themselves the risk of exposure. Suppose each hidden method has a corresponding logo S and the attacker has a logo set Φ which consists of some hidden methods' logos. Once he find the logo S which matches a logo in Φ, he can easily recognize the very method. To solve this problem, we propose a method based on synchronization to hide the specific sign or logo. First, the sender generates a key using a public variable parameter which is always changing from time to time. Then, we can calculate the hidden data's location in the cover from the key. According to the locations, we can embed the logo into the cover. As the public parameter is changing from time to time, each transmission of hidden data has a unique location sequence. When the stego reaches its receiver, according to the public parameter, the receiver could generate the key and get the hidden data's location to extract the secret data correctly. Experimental results verify that the data hiding method performs well and hardly has impacts on the cover's quality and has little impacts on the robustness, imperceptibility and capacity of the original stego-cover. Besides, it is able to recover the key with linear time complexity when the critical information which is used to generate the key is missing.

Key words: Data hiding, automatically changing, chaotic encryption, synchronization

INTRODUCTION

Data hiding techniques are mainly about researching on hiding data into cover. Avoiding being noticed by adversaries is their main purpose. Until now, it consists of Steganography, Digital Watermark, Visual Cryptography and etc. With data hiding techniques, secret communications become possible. Data hiding techniques are being applied to military, business, finance areas, such as conveying military intelligence, the privacy data of e-business, secret contracts between business partners and so on. Besides, data hiding techniques are utilized for identification, copyright protection, integrity authentication and so on.

Nowadays, main researches on data hiding techniques can be roughly divided into three areas: Spatial domain, transform domain and file format.

For spatial domain methods, LSB and LSB matching are the most commonly used algorithms. Researches on data hiding in spatial domain focus on capacity improvement, imperceptibility advancement and resistance against steganalysis (Zhang, 2011; Omoomi *et al*., 2011; Geetha *et al*., 2011; Zeki *et al*., 2011; Zanganeh and Ibrahim, 2011; Rajagopalan *et al*., 2014).

For transform domain methods, though data hiding methods are more complicated, they often achieve better robustness than spatial domain ones. The QIM and adaptive embedding algorithms (Wang *et al*., 2002; Podilchuk and Zeng, 1998) are typical of these methods. There are also some researches (Cancellaro *et al*., 2011; Lian *et al*., 2007; Khan *et al*., 2011; Bouslimi *et al*., 2012) on techniques about reversible watermarking of encrypted media. Some (Qiao and Nahrstedt, 1998; Shi and Bhargava, 1998; Wu and Kuo, 2005; Qiu and Wang, 2012) are on encryption techniques for the compressed image and some (Chen *et al*., 2003; Abdulfetah *et al*., 2010) focus on robustness and imperceptibility.

Style of file (such as XML, HTM, PDF, etc.) can be utilized in data hiding as well (Bo *et al*., 2009; Shahreza, 2006; Cantrell and Dampier, 2004; Liu and Tsai, 2007; Liu *et al*., 2008), according to the character of the file's style. Such techniques are based on specific style of files which are different from each other.

In some cases, when extracting hiding information, we always needs a logo to mark the specific steganography algorithm. This brings great unsafety to the hiding system. For example, the software ffencode (Schroder, 2005) embeds a fixed sequence of

Corresponding Author: Gang Luo, College of Information Science and Engineering, Hunan University, Changsha, 410082, China

"01100110011001100110001101101111011001000110010 1" into the carrier without any hiding processing. As long as an attacker calculates this specific sequence, he or she can confirm that the carrier he or she catches is encrypted by ffencode. Unfortunately, there are few researches on how to resolve such problems. Our purpose is to resolve this kind of problems. We put forward a method that can make the logo's appearance different from time to time. Meanwhile, it makes sure that the receiver can extract the logo with 100% accuracy and recognizes the steganography tool correctly. And when the key is missing, it can recover the key with limited time cost in linear time complexity.

METHODOLOGY

Problem statements and analysis: To hide the specific logo, we need to encrypt it first. However, a proficient can get the logo with Chosen-plaintext attack when the logo is encrypted by the same key all the time. To avoid this, we need a key that can automatically change itself. A changing mechanism is what we need. Using an automatically changing key, the key of the sender's must synchronize with the receiver's. When sender's key doesn't match the receiver's, we need a mechanism to restore the key's synchronization. Considering all the requirements listed above, we take system time as the changing factor of the key. The system time has 4 key important natures: Linearity, uniqueness, finiteness in a finite interval and dynamics. The uniqueness and dynamics nature of time are necessary to the key's automatically changing. And in case of being damaged or tampered, we need an efficient way to recover the key. The finiteness and linearity nature of the time can be used to recover the key. We also need a method to hide the encrypted logo. Comparing to hiding a logo in a fixed position, it is safer to hide the logo in random positions. The chaotic system performs well in producing pseudo-random sequences. Among chaotic mappings, the skew tent mapping has excellent pseudo-randomness and it obeys uniform distribution. With the skew tent mapping, we calculate the embedding locations so that it has the ability to resist statistical analysis.

Merits of the method:

- Choosing different keys from time to time
- Being able to figure out the missing or damaged key in linear time complexity
- Resistance to statistics analysis

Description of the method: First, P stands for plaintext, VI denotes the initial vector of 128 bits. T is the time that generated by the computer system and its precision is second. In this study, we choose the format of time as "YYYYMMDDHHMISS". Specifically, we encrypt the time T with MD5 algorithm (128 bits) and get a sequence of hex numbers: $T_{md5} = MD5(T)$, $T_{md5} = t_{md5\_0}t_{md5\_1}\ldots t_{md5\_15}$. S is the logo that need to be encrypted and its length is l bytes (each character is 1 byte and 1 byte = 8 bits). VI and S are divided into the following groups:

$$VI = vi_0vi_1vi_2\ldots\ldots vi_{15}$$

$$S = s_0s_1\ldots\ldots s_{l-1}$$

(Each vi_i or s_i is 8 bits).

According to T_{md5} and VI, the system generates the secret key K in the following way:

$$K = VI \oplus T_{md5}$$

$$k_i = vi_i \oplus t_{md5\_i},\ i \in [0, 15]$$

The logo S is encrypted to M in the following way:

$$m_i = k_{(i \bmod 16)} \oplus s_i = b_{m\_8i}b_{m\_8i+1}b_{m\_8i+2}\ldots b_{m\_8i+7},\ i \in \{0, 1, 2\ldots l\text{-}1\}$$

$$M = m_0m_1m_2\ldots m_{l-1} = b_{m\_0}b_{m\_1}b_{m\_2}\ldots b_{m\_8l-1},\ b_{m\_i} \in \{0, 1\}$$

The M is what we need to hide into the carrier. Its length is the same with S, the length of M is also l bytes. And $T^{(j)}$ is the jth created time and the corresponding secret key is $K^{(j)}$ (j is an integer). T is different from time to time and the $K^{(j)}$ is created by $T^{(j)}$ nd VI. When $i \neq j$, $K^{(i)} \neq K^{(j)}$.

Logistic map is often used to design random encryption system. Logistic mapping not only has low computing complexity but also has good pseudo-random performance. However, the skew tent mapping's distribution is more uniform than logistic mapping's (Xiang *et al*., 2008) and it has stronger pseudo-randomness as well (Masuda and Aihara, 2001). Therefore, the chaotic sequence generated by the skew tent mapping is better for cryptographic purpose:

- Logistic mapping:

$$x_{i+1} = u \times x_i \times (1\text{-}x_i),\ u \in [0, 4],\ x \in (0, 1) \quad (1)$$

- Skew tent mapping:

$$f(x) = \begin{cases} x/p, & x \in [0, p) \\ (1\text{–}x)/(1\text{–}p), & x \in [p, 1] \end{cases} \quad (2)$$

We set r (r is 8 bits) as follows:

$$r = (k_0 \oplus k_1 \oplus k_2 \ldots \oplus k_{15}) \bmod 16$$

To get the hidden data's location by skew tent map, x_0, p, N is initiated as follow:

$$V_L = k_{r \bmod 16} k_{(r+1) \bmod 16} \ldots k_{(r+7) \bmod 16}$$
$$V_R = k_{(r+8) \bmod 16} k_{(r+9) \bmod 16} \ldots k_{(r+15) \bmod 16}$$

$$U_L = k_0 k_1 \ldots k_7 \quad U_R = k_8 k_9 \ldots k_{15}$$

$$x_0 = (V_L \oplus V_R)/2^{64}, \; P = (U_L \oplus U_R)/2^{64}, \; N = k_0 \oplus k_1 \oplus k_2 \ldots \oplus k_{15}$$

We embed the encrypted logo M into a cover grayscale I, M is l bytes and the length of the cover I is L bytes. Suppose there is a logo set $\Phi = \{S_0, S_1, S_2 \ldots S_{n-1}\}$ and the $l_{div} = \max\{\text{length}(S_i)\}+\varepsilon$, $\varepsilon+N$ and ε is agreed on by both the sender and receiver. The cover I is divided into $8l_{div}$ ($l_{div}>l$) blocks. There are $g(g = \lfloor L/8l_{div} \rfloor)$ pixels in each block and the block set is denoted as Block = $\{B_0, B_1, B_2 \ldots B_{8l_{div}-1}\}$ The ith bit of $M(b_{m\_i})$ is embedded into the block B_i. To embed the bit b_{m_i} into the block B_i, we set $d_i = \lfloor g \times x_i \rfloor$ as the embedding location of B_i and embed b_{m_i} into the d_ith pixel of B_i ($0 \le i \le 8l_{div}-1$). Then we convert pixel's value from decimal to binary, in the form of $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ (0~255). To hide the bit b_{m_i}, if $b_0 = b_{m\_i}$, we do nothing to b_0, otherwise we replace b_0 with b_{m_i}.

Figure 1 shows the embedding process and the embedding algorithm is as follows:

Input: T is system time, VI is initial vector, 8 L is the bit length of cover I and it is divided into $8l_{div}$ blocks, M is the encrypted form of S, its length is 8l bits.
Output: $d_0, d_1, d_2 \ldots d_{8l_{div}-1}$
Algorithm:

Step 1: $K = (MD5(T) \oplus VI)$
Step 2: $M = \cup(s_i \oplus k_{(i \bmod 16)})$, $i \in \{0, 1, 2 \ldots l-1\}$
Step 3: $r = (k_0 \oplus k_1 \oplus k_2 \ldots \oplus k_{15}) \bmod 16$
If (r = = 8) change T and go to step 1.

$$V_L = k_{r \bmod 16} k_{(r+1) \bmod 16} \ldots k_{(r+7) \bmod 16}$$
$$V_R = k_{(r+8) \bmod 16} k_{(r+9) \bmod 16} \ldots k_{(r+15) \bmod 16}$$
$$U_L = k_0 k_1 \ldots k_7 \; U_R = k_8 k_9 \ldots k_{15}$$
$$x_0 = (V_L \oplus U_R)/2^{64}$$
$$P = (U_L \oplus U_R)/2^{64}$$

Step 5: For i = 0 to 8l-1

$$x_{i+1} = \text{Tent}(x_i, p, N)$$

Step 6: $g = \text{floor}(L/8l_{div})$
Step 7: For j = 0 to 8l-1

$$d_j = \text{floor}(g \times x_j)$$

Step 8: For i = 0 to 8l-1

$$loc_i = g \times i + d_i$$

Embedding the ith bit b_{m_i} of M into the loc_ith pixel of I

The sender embed M into I and the original I becomes stego-carrier I_m. The receiver already has VI and receives T from the sender. During the extraction phase, he generates K with the T and VI. The receiver gets the sub-location sequence $d_0 d_1 d_2 \ldots d_{8l_{div}-1}$ to extract the message M_M which contains M. With the K, he decrypts M_M and gets the message S_S which contains logo S.

Figure 2 shows the extracting process and the extracting algorithm is as follows:

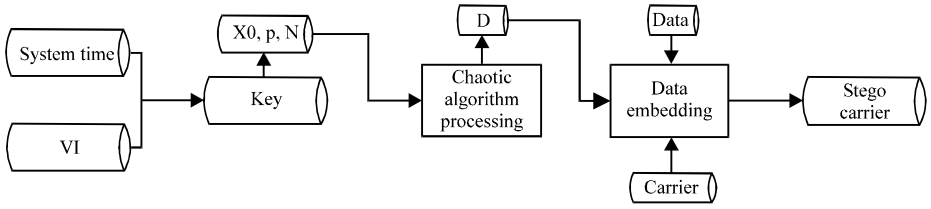

Fig. 1: Embedding process

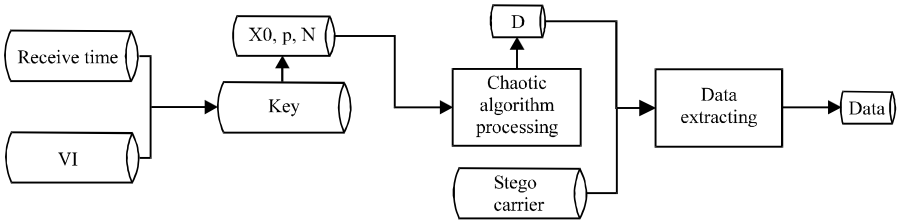

Fig. 2: Extracting process

Input: T is received from the sender, VI is initial vector, 8L is the bit length of stego-carrier I_m
Output: S
Algorithm:

Step 1: $K = (MD5(T) \oplus VI)$
Step 2: $r = (k_0 \oplus k_1 \oplus k_2 \ldots \oplus k_{15}) \bmod 16$
If $(r == 8)$ change T and go to step 1.

$$V_L = k_{r \bmod 16} k_{(r+1) \bmod 16} \ldots k_{(r+7) \bmod 16}$$

$$V_R = k_{(r+8) \bmod 16} k_{(r+9) \bmod 16} \ldots k_{(r+15) \bmod 16}$$

$$U_L = k_0 k_1 \ldots k_7 \; U_R = k_8 k_9 \ldots k_{15}$$

$$x_0 = (V_L \oplus V_R)/2^{64}$$

$$P = (U_L \oplus U_R)/2^{64}$$

$$N = k_0 \oplus k_1 \oplus k_2 \ldots \oplus k_{15}$$

Step 3: For $i = 0$ to $8l_{div}-1$

$$x_{i+1} = Tent(x_i, p, N)$$

Step 4: $g = floor(L/8l_{div})$
Step 5: For $j = 0$ to $8l_{div}-1$

$$d_j = floor(g \times x_j)$$

Step 6: For $i = 0$ to $8l_{div}-1$

$$loc_i = g \times j + d_i$$

Step 7: Extract M_M from I_m. (M_M is $8l_{div}$ bits)
Step 8: $S_S = Decrypt(M_M)$
if $(S_S \supseteq S)$ Extracting Success.
else recover T

To extract the hidden logo S, the extracting algorithm needs to extract $8l_{div}$ bits from I_m and get the message M_M which is l_{div} bytes. It decrypts M_M to the form of $S_S = S \cup S_e$, $S_e = s_l s_{l+1} \ldots s_{8l_{div}-1}$. We compare S_S with each logo in the set Φ. If we find that there is a logo $S_i \in \Phi$ and $S_i = S$, then we get the correct logo.

Analysis of the algorithm: Both the sender and receiver need to have the same logo set Φ. The algorithm embeds an uncertain logo that is from the logo set Φ into the cover I. Because the receiver doesn't know which logo the sender selects, to define a fixed length l_{div} is necessary. In this study, l_{div} meets the following requirements: $l_{div} \geq \max\{l | l = length(S), S \in \Phi\}$ Adding a new logo to the set Φ needs to reset l_{div}. There is a difference between embedding and extracting: The T is generated by the system at the sender side while the receiver receives T instead of generating it. And the using of skew tent mapping makes sure that the logo bits are uniformly hidden in the cover. With a dynamic T, the hidden locations are different from time to time.

In addition, when T_c (system time) is damaged or tampered by some attacker, we still have a chance to obtain it. Suppose we lost T_c and we have T_{last} which is got from the last transmission. And we received the M at the real time T_{real}. ($T_c \in (T_{last}, T_{real})$ T here is a integer). We can try each T from either T_{last} to T_{real}, or T_{real} to T_{last}. It is certain that we can deduct the correct T which is equal to T_c. And if the searching direction is from T_{last} to T_{real} we need to set the time cost to Δt_{last}. And if the time interval is from T_{real} to T_{last}, we need to set the time cost as Δt_{real}. In addition, we denote the time cost to locate T_c as Δt and $\Delta t \leq \max\{\Delta t_{last}, \Delta t_{real}\}$. And the searching of T_c is in a linear way, so the time complexity to find T_c is $O(\Delta t)$.

Experiments: Let T_c = 20140303122212, VI = 7154A05BCC 3E552D5E5E21BA716A31EF and the images size be 512×512, 512×256 and 256×256. Each image size consists of 456 images. The images we use for experiments are all from http://sipi.usc.edu/database/. Let secret logo be m. Set m = "Hunan University is a good university". We choose LSB embedding method for the experiments, because it is easy to operate and causes little distortion to the original images.

Figure 3 shows the impacts on 1000 locations when T_c is changing from T_c = 20140303122212 to T_s = 20140303122213. y is the degree of location's difference between T_c and T_s. d_{c_i} is the ith location generated by the T_c, d_{s_i} is the ith location generated by T_s. y_i is the degree of difference between d_{c_i} and d_{s_i}. $y_i = |d_{c\_i} - d_{s\_i}|/8l$, 8l is the bit length of S, $i \in \{0, 1, 2 \ldots 8l-1\}$. The average degree of difference from 8l locations is:

$$\bar{y} = \frac{\sum_{i=0}^{8l-1} y_i}{8l}$$

In this experiment, we calculate the $\bar{y}$ = 33.99%.

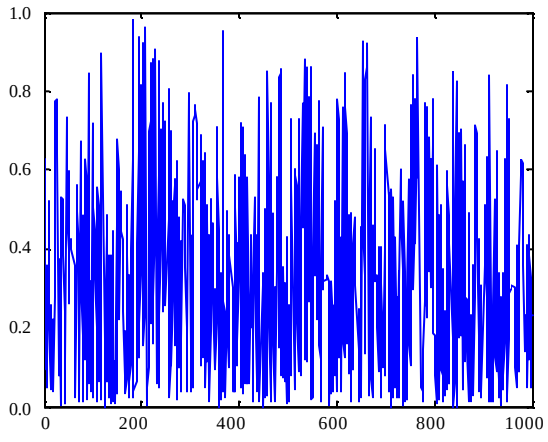

Fig. 3: Key sensitivity to initial value

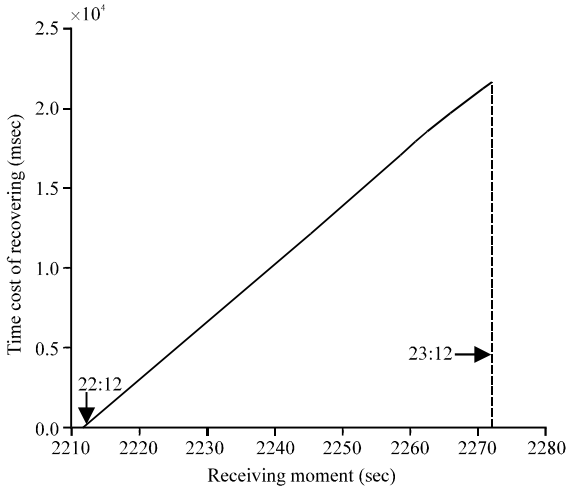

Fig. 4: Cost time for T's recovery

Table 1: Average of MSE and PNSR

Image size (pixel)	$\overline{MSE}$	$\overline{PSNR}$ (dB)
256×256	6.57e-04	79.95
512×256	3.279-04	82.97
512×512	1.639e-04	85.98

Here, we count the changed locations and unchanged ones. When the ith location is changed, then $z_i = 1$, otherwise, $z_i = 1$. Then we calculate the changing rate of the locations when time changes from T to T_s. We set rc as the global changing rate:

$$rc = \frac{\sum_{i=0}^{8l-1} z_i}{8l}$$

Testing 456 images, we find the average rc is $\overline{rc} = 99.758\%$.

Figure 4 shows the time cost from different starting position when T_c = 22:12 is the correct solution. And from the figure, we can see that the time cost to find the exact $T = T_c$ is linear. $\Delta t = |T_x - T_i|$, the value of the Δt is determined by T_x. And the longer the logo is, the steeper the line is. It proved that the searching for the missing key cost is $O(\Delta r)$.

Generally, a logo is short, embedding it hardly causes distorsion to the cover. There are 3 kinds of image size, each size consists of 456 images. According to Table 1, we could easily find that, the $\overline{MSE}$ of each size is very close to 0 and the $\overline{PSNR}$ has a high value that is more than 80. It verifies the fact that embedding the logo has little impacts on original algorithm's robutness, capacity and invisibility. And we could also conclude from the Table 1 that, the larger the image size is, the less impacts caused by the embedding.

DISCUSSION

In this study, we analyzed the feasibility of the algorithm. In the process of embedding, we encrypt logo with a dynamic key. Then embed the encrypted logo into cover in a random way. In this way, the attacker has little chance to find the logo. When in extracting process, we need to extract an encrypted message that contains the encrypted logo. We decrypt the encrypted message and get the message which contains logo. In this way, we can use a set of data hiding algorithms which is based on logo, instead of only one. By conducting experiments on testing how sensitive the key to the chaotic mapping's parameter, we find that the hiding location sequence's ($d_1d_2d_3\ldots$) changing rate is close to 100% when T changes 1 bit. When the time T_c for generating key is damaged, the algorithm can recover the damaged T_c in a linear computation complexity with limited cost. By embedding logos into more than 1300 images of different sizes, measuring by both the MSE and PSNR shows that the algorithm has little impact on the robutness, capacity and invisibility of original one. However, though the proposed algoritm can resolve the risk but there is still an efficient problem. The system time T for K's generation is not hidden during the message's transmission, it could be tampered easily. Though there is a way for T's recovery, it degrades the efficiency of the algorithm to some extent.

CONCLUSION

The data hiding software which relies on some fixed logo to identify themselves can perform data hiding and data extracting efficiently. However, the fixed logo may bring great risk to the data hiding algorithm, because it can be easily caught by an attacker especially when the attacker has knowledge of the algorithm. To solve this problem, this study propose a method to enable the logo to change itself automatically by adding time variable. In theory, each change of the logo is different from each other. And the method hides the logo bit by bit with a random sequence which is generated by the skew tent mapping. The random sequence is used to calculate the bit's hiding location. In this way, the attacker can hardly extract the logo even though he has the knowledge of the data hiding algorithm. Furthermore, when the time is damaged, there is a remedy for the receiver to generate the key in limit time cost. Though the proposed method resolves the problems brought by the fixed logo, there is still room for improvements. The future study will focuse on improving the robustness of the algorithm. Besides, we need to find an efficient way to protect the system time which is exposed during the transmission.

ACKNOWLEDGMENTS

This study was supported by National Natural Science Foundation of China (No. 61202439, 61103215) and the Fundamental Research Funds for the Central Universities of China.

REFERENCES

Abdulfetah, A.A., X. Sun, H. Yang and N. Mohammad, 2010. Robust adaptive image watermarking using visual models in DWT and DCT domain. Inform. Technol. J., 9: 460-466.

Bo, L., L. Wei, C. Yuan-Yuan, J. Dong-Dong and C. Ying-Zhi, 2009. HTML integrity authentication based on fragile digital watermarking. Proceedings of the IEEE International Conference on Granular Computing, August 17-19, 2009, Nanchang, China, pp: 322-325.

Bouslimi, D., G. Coatrieux and C. Roux, 2012. A joint encryption/watermarking algorithm for verifying the reliability of medical images: Application to echographic images. Comput. Methods Programs Biomed., 106: 47-54.

Cancellaro, M., F. Battisti, M. Carli, G. Boato, F.G.B. De Natale and A. Neri, 2011. A commutative digital image watermarking and encryption method in the tree structured Haar transform domain. Signal Process.: Image Commun., 26: 1-12.

Cantrell, G. and D.D. Dampier, 2004. Experiments in hiding data inside the file structure of common office documents: A steganography application. Proceedings of the International Symposium on Information and Communication Technologies, June 16-18, 2004, Las Vegas, Nevada, USA., pp: 146-151.

Chen, T.H., G. Horng and S.H. Wang, 2003. A robust wavelet-based watermarking scheme using quantization and human visual system model. Inform. Technol. J., 2: 213-230.

Geetha, S., V. Kabilan, S.P. Chockalingam and N. Kamaraj, 2011. Varying radix numeral system based adaptive image steganography. Inform. Process. Lett., 111: 792-797.

Khan, M.I., V. Jeoti, A.S. Malik and M.F. Khan, 2011. A joint watermarking and encryption scheme for DCT based codecs. Proceedings of the 17th Asia-Pacific Conference on Communications, October 2-5, 2011, Sabah, Malaysia, pp: 816-820.

Lian, S., Z. Liu, Z. Ren and H. Wang, 2007. Commutative encryption and watermarking in video compression. IEEE Trans. Circuits Syst. Video Technol., 17: 774-778.

Liu, T.Y. and W.H. Tsai, 2007. A new steganographic method for data hiding in microsoft word documents by a change tracking technique. Trans. Inform. Forensics Secur., 2: 24-30.

Liu, Y., X. Sun, Y. Liu and C.T. Li, 2008. MIMIC-PPT: Mimicking-based steganography for microsoft power point document. Inform. Technol. J., 7: 654-660.

Masuda, N. and K. Aihara, 2001. Cryptosystems based on space-discretization of chaotic maps. Proceedings of the IEEE International Symposium on Circuits and Systems, Volume 3, May 6-9, 2001, Sydney, Australia, pp: 321-324.

Omoomi, M., S. Samavi and S. Dumitrescu, 2011. An efficient high payload ±1 data embedding scheme. Multimedia Tools Appl., 54: 201-218.

Podilchuk, C.I. and W. Zeng, 1998. Image-adaptive watermarking using visual models. IEEE J. Sel. Areas Commun., 16: 525-539.

Qiao, L. and K. Nahrstedt, 1998. Comparison of MPEG encryption algorithms. Comput. Graphics, 22: 437-448.

Qiu, J. and P. Wang, 2012. Encryption algorithm for compressed image based on chaotic maps. Comput. Sci., 6: 44-46.

Rajagopalan, S., H.N. Upadhyay, S. Varadarajan, J.B.B. Rayappan and R. Amirtharajan, 2014. Gyratory assisted info hide-a nibble differencing for message embedding. Inform. Technol. J., 13: 2005-2010.

Schroder, B., 2005. Ffencode for DOS. http://www.burks.de/stegano/ffencode.html

Shahreza, M.S., 2006. A New Method for Steganography in HTML Files. In: Advances in Computer, Information and Systems Sciences and Engineering, Elleithy, K., T. Sobh, A. Mahmood, M. Iskander and M. Karim (Eds.). Springer, Netherlands, ISBN-13: 9781402052606, pp: 247-252.

Shi, C. and B. Bhargava, 1998. A fast MPEG video encryption algorithm. Proceedings of the 6th ACM International Conference on Multimedia, September 13-16, 1998, Bristol, UK., pp: 81-88.

Wang, S., X. Zhang and T. Ma, 2002. Image watermarking using dither modulation in dual-transform domain. J. Imaging Soc. Jpn., 41: 398-402.

Wu, C.P. and C.C.J. Kuo, 2005. Design of integrated multimedia compression and encryption systems. IEEE Trans. Multimedia, 7: 828-839.

Xiang, T., K. Wong and X. Liao, 2008. An improved chaotic cryptosystem with external key. Commun. Nonlinear Sci. Numer. Simul., 13: 1879-1887.

Zanganeh, O. and S. Ibrahim, 2011. Adaptive image steganography based on optimal embedding and robust against chi-square attack. Inform. Technol. J., 10: 1285-1294.

Zeki, A.M., A.A. Manaf and S.S. Mahmod, 2011. High watermarking capacity based on spatial domain technique. Inform. Technol. J., 10: 1367-1373.

Zhang, X., 2011. Reversible data hiding in encrypted image. IEEE Signal Process. Lett., 18: 255-258.