Journal of Computer Science 2 (2): 148-153, 2006
ISSN 1549-3636
© 2006 Science Publications

Application of T-Code, Turbo Codes and Pseudo-Random Sequence for Steganography

Anil Kumar and Navin Rajpal
School of Information Technology, Guru Gobind Singh Indraprastha University
Kashmere Gate, Delhi, India

Abstract: In this study, we propose new technique that will address the problem of robustness and data safety in steganography. The steganography consists of techniques to allow the communication between two persons, hiding not only the contents but also the very existence of the communication in the eyes of any observer. T-Codes used with Turbo Codes generates cryptic and error-coded data stream, which is hidden in the stego-object using Pseudo-Random sequence. This technique makes our processed data stream non-vulnerable to the attack of an active intruder, or due to noise in the transmission link.

Key words: T-code, turbo codes, data hiding, steganography, pseudo-random sequence

INTRODUCTION

In the present era of computers and fast communication, one needs to protect communicated information (message or plain text) from unauthorized user, while sending it through any electronic media. One such technique to protect the data is Steganography. The Steganography consists of techniques to allow the communication between two persons. It hides not only the contents but also the existence of the communication in the eyes of any observer. These techniques use a second perceptible message, with meaning disjoined by the secret message. This second message works as a "Trojan horse" and is a container of the first one[1,2]. The new technologies and in special way, the information networks require more and more sophisticated strategies in order to prevent the message privacy. In this context, digital images and audio is excellent candidate to turn into containers of the messages, since the bits of a secret text message can be superimposed, as slight noise, to the bits employed for coding a digital image. Historically, the first instance of the use of Steganography is found when the Greeks received warning of Xerxes hostile intentions from a message underneath the wax of a writing tablet[3]. The Chinese practiced Steganography by embedding a code ideogram at a prearranged place in a dispatch while medieval Europe utilized grill systems in which a paper or wooden template would be placed over a seemingly innocuous text, highlighting an embedded secret message. The scientific study of Steganography can be traced to Simmons who in 1983 formulated it as the "Prisoners Problem". In this scenario, two prisoners (Alice and Bob) wish to devise an escape plan. However, all there communications pass through the warden (Willie) and if he detects any encrypted messages he will frustrate their plan by throwing them into solitary confinement. So they must find some way of hiding their cipher text in an innocent looking cover text. Also they must ensure their plan does not get destroyed through transmission channel.

By definition, T-codes are variable-length codes obtained by applying the prescribed T-augmentation algorithm[4]. All T-codes exhibit strong tendency towards self-synchronization by virtue of the construction algorithm. However, some T-codes exhibit stronger synchronization performance than others. Two important results that have emerged from the investigation of T-codes can be summarized as follows[5,6]:

i. T-codes that show the best synchronization performance are also among the most efficient.
ii. Within any given subgroup of T-codes, there is a relatively wide band of different synchronization performances. For example, T-codes that are among the most efficient for an information source may require anything between 1.5 to 3 characters to attain synchronization following a lock loss.

Turbo-codes were discovered in 1993[7-11]. Turbo-codes promise good results in the error-correction techniques, sought for nearly half a century: to achieve the ultimate limits of capacity of a communication channel. It is not surprising, therefore, that they have very rapidly moved from the research laboratories to find practical application in communication field mainly in wireless communication. We shall introduce the two concepts that are the basis of turbo-codes: concatenated coding and iterative decoding. These are the real secret of turbo-codes and also the basis of their name. But will the discovery of turbo codes put an end to the story of error control coding? It might be expected that the discovery of the ultimate codes would make any further work unnecessary, but in fact the reverse seems to be the case: it has resulted in a

Corresponding Author: Anil Kumar, School of Information Technology , Guru Gobind Singh Indraprastha University, Kashmere Gate,Delhi, India

renaissance of coding research, both in universities and in industry world-wide.

There are two methods of performing steganography, one in spatial domain and the other in frequency domain. Each technique has its own advantage and disadvantage. In the spatial domain[12,13], we can simply insert data into host image by changing the gray levels of some pixels in the host image, but the inserted information may be easily detected using computer analysis. In the frequency domain[14,15], we can insert data into the coefficients of a transformed image, for example using discrete Fourier transform (DFT), discrete cosine transform (DCT) and discrete Wavelet transform (DWT). But we cannot embed too much data in the frequency domain because the quality of the host image will be distorted significantly.

T-CODES

T-code is a subset of all possible Huffman code sets[4]. T-codes are used for compression. The T-codes are self-synchronizing, so if some bits are lost or modified in a T-code encoded stream, the decoder will regain synchronization automatically. We build a T-code set to understand the process. A simple T-code set consists of the alphabets. With a binary alphabet, this is $S=\{0,\ 1\}$. We then remove one of the elements of the set and use it as a prefix to extend the initial set so that we get more codes. Let us take an example, let us use the first element 0 as the prefix. The new code set therefore is $S(0)=\{1,\ 00,\ 01\}$. For the next level, if we use 1 as the prefix, we get the set $S(0,\ 1)=\{00,\ 01,\ 11,\ 100,\ 101\}$; and if we use 00 as the prefix, we would get the set $S(0,\ 00)=\{1,\ 01,\ 001,\ 0000,\ 0001\}$; and if we use 01 prefix, then the resultant set $S(0,\ 01)=\{1,\ 00,\ 011,\ 0100,\ 0101\}$. Short code words make better prefixes, for they result in shorter codes in the resulting code set. This result gives the maximum compression. The total number of elements in the T-code set at T-augmentation level n is 2n+1. Using this property we can to find out the T-augmentation levels required for encoding m symbols.

Self synchronization: T-codes self-synchronize during decoding, so if some bits are lost or modified in a T-code encoded stream, the decoder will regain synchronization automatically. Let us take an example. Consider the message helloworldx that contains 8 different characters *{h, e, l, o, w, r, d, x}* and with frequencies {1, 1, 3, 2, 1, 1, 1} respectively. Encoding this message requires constructing a T-code set with T-augmentation level 3. Using short codes as prefixes at each T-augmentation level, we arrive at the T-code set S(0, 1, 0)={01, 11, 100, 0000, 0001, 00100, 00101}. Assigning short codes to those characters with high frequencies, we get the following dictionary:

Table 1: Dictionary generated for helloworldx

Character	H	E	L	O	W	R	D	X
Code	100	101	01	11	0000	0001	00100	00101

With this dictionary, the string helloworldx will be encoded as 100.101.01.01.11.0000.11.0001.01.00100.00101. Typical errors one may encounter while decoding a bit stream are bit losses and inversions. Let us examine how the bit stream representing helloworldx will be decoded in each of these cases.

Bit loss: Assume that the two underlined bits in 100.101.01.01.1$\underline{1}$.0000.1$\underline{1}$.0001.01.00100.00101 are destroyed or missing. The bit stream will then be decoded as 100.101.01.01.$\underline{100.00100.01}$.01.00100.00101, or *hell$\underline{hrl}$ldx*, where underlining shows the errors.

Bit inversion: Assume that the two underlined bit in 100.101.01.01.11.000$\underline{0}$.1$\underline{1}$.0001.01.00100.00101 inverted in the bit stream 100.101.01.01.11.000$\underline{1}$.1$\underline{0}$.0001.01.00100.00101. The bit stream will then be decoded as 100.101.01.01.11.$\underline{0001.100.00101}$.00100.00101, or *hello$\underline{rhx}$dx,* where the underlining shows the errors.

Note that if the message is large enough and the bit errors are small enough, a considerable portion of the message will be correctly decoded. This is the main advantage of this T-code. This is the main advantage of The T-Code

TURBO CODES

The power of Forward Error Correction (FEC) codes increases with length *k* and approaches the Shannon bound[11,16] only at very large *k*, but also that decoding complexity increases very rapidly with *k*. Therefore it would be desirable to build a long, complex code out of much shorter component codes, so that it can be decoded much more easily. Concatenation provides a very straightforward means of achieving this (Fig. 1).

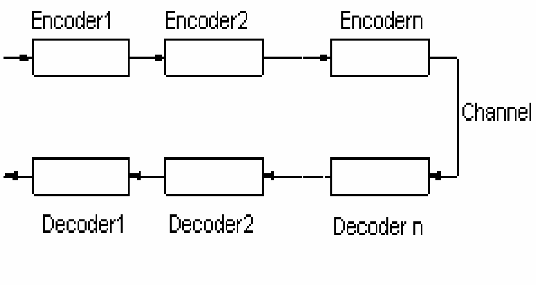

Fig. 1: Principle of concatenated codes

In this feed the output of one encoder (the *outer* encoder) to the input of another encoder and so on, as required. The final encoder before the channel is known as the *inner* encoder. The resulting composite code is clearly much more complex than any of the individual codes, which is the desirable for the decoding. However it can readily be decoded: we simply apply each of the component decoders in turn, from the inner to the outer. This simple scheme suffers from main drawback, which is called *error propagation*. If a decoding error occurs in a codeword, it usually results in a number of data errors. This is cascade in the nature. When these are passed on to the next decoders they may or may not able correct the errors. The performance of the outer decoder improved if these errors were distributed between a numbers of separate code words. For this purpose we are using an inter-leaver /de-inter-leaver. The inter-leaver is illustrated in Fig. 2:

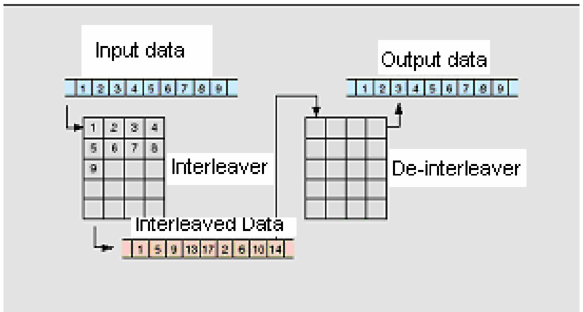

Fig. 2: Operation of interleaver and de-interleaver

This simple inter-leaver (*rectangular* or *block* inter-leaver) consists of a two-dimensional array, into which the data is read along its rows. Once the array is full, columns, thus permuting the order of the data, read out the data. (An inter-leaver is commonly denoted by the Greek letter π and its corresponding de-inter-leaver by π^{-1}.) The original order can then be restored by a corresponding de-inter-leaver: an array of the same dimensions in which the data is read in by columns and read out by rows. This inter-leaver placed between the outer and inner encoders of a concatenated code and a de-inter-leaver between the inner and outer decoders in the receiver, as shown in Fig. 3.

Hence, provided the outer code is able to correct at least one error, it can always cope with single decoding errors in the inner code. Usually the block codes used in such a concatenated coding scheme are *systematic*: that is, the g data bits appear in the codeword, along with *n-g parity* or *check* bits. Now suppose the outer code has data length g1 and code length *n*1, while the inner code has data length g2 and code length *n*2 and the inter-leaver has dimension *g*2 rows by *n*1 columns. Then the parity and data bits may be arranged in an array as shown in Fig. 4.

Iterative decoding: The 'Turbo' principle: The decoding technique shown in Fig. 3 the inner code is decoded first and then we decode the outer one.

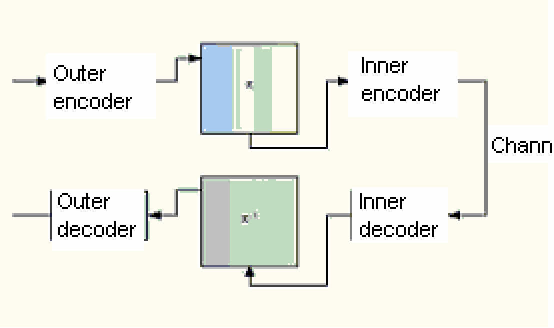

Fig. 3: Concatenated encoder and decoder with inter-leaver

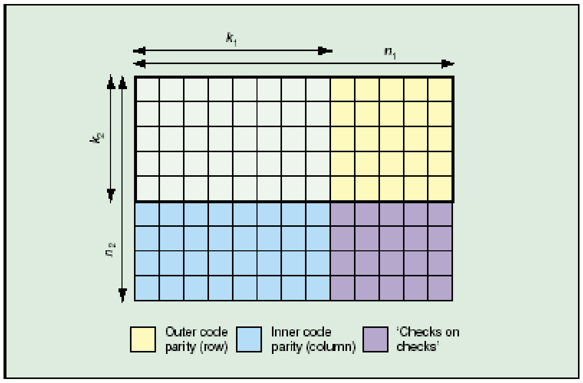

Fig. 4: Array for interleaved concatenated code

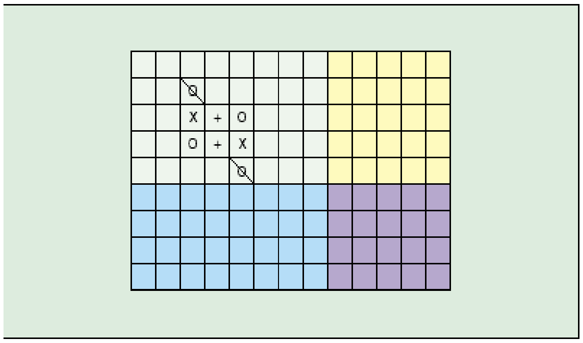

Fig. 5: Pattern of received errors ('O') in Codeword array, with errors introduced by inner (column) decoder ('X') and outer (row) decoder ('+')

However, this may not always be as effective as we hope about it. Consider a received codeword array with the pattern of errors shown by the 'O's in Fig. 5.

Suppose that both component codes are capable of correcting single errors only. As mentioned above, if there are more errors than this the decoder produces cascading error effect. For the pattern shown this is the case for two of the column code words and errors might be added as indicated by 'X'. When this is applied to the outer (row) decoder some of the original errors may be corrected (indicated by a cross through the 'O'), but yet more errors may be inserted (marked with '+'). The recovery depends on how we decode the data.

Therefore, the original pattern would have been decoded correctly had it been applied to the row decoder first. Therefore this is the basis of the iterative decoder: to reapply the decoded words to the outer and repeat as many times until the error are reduce below the threshold point. One further improvement is required for the iterative decoder. That known as *soft-in, soft-out* (SISO) decoding as shown in Fig 6.

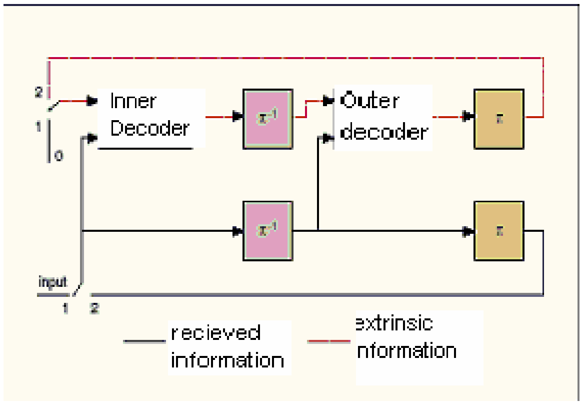

Fig. 6: Iterative decoder. The switches are in position 1 for the first iteration and position for subsequent iterations

This is well known that the performance of a decoder is significantly enhanced if we use, in addition to the 'hard decision', some additional 'soft information' on the reliability of that decision is passed to the decoder. Usually a fixed number of iterations are used—between 4 and 15, depending on the type of code and its length—but it is also possible to detect convergence and terminate the iterations at that point.

PSEUDO RANDOM CODE

A Non-Linear forward feedback shift Register (NLFFSR) is a mechanism for generating Pseudo random binary sequences[17-20]. Figure 7 shows a general model of an n-bit NLFFSR. It is a Non-linear forward feedback shift register with a feedback function f

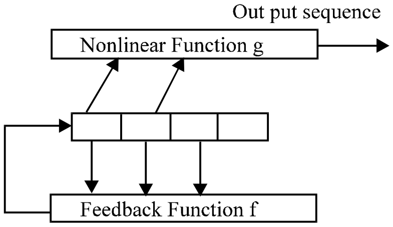

Fig. 7: A general model of 4-bit NLFFSR

NLFFSR are extremely good pseudorandom binary sequence generators[17-20]. When this register is loaded with any given initial value (except 0 which will generate a pseudorandom binary sequence of all 0s) it generates pseudorandom binary sequence, which has very good randomness and statistical properties. The only signal necessary for the generation of the binary sequence is a clock pulse. With each clock pulse a bit of the binary sequence is produced. A model of 4-bit NLFFSR is considered to demonstrate the functioning of NLFFSR with the feedback function $f = 1 + x + x^4$ and the non-linear function g defined by $a_{n-1}.a_{n-3} \oplus a_{n-2}.a_{n-4}$ forming non-linear feed forward shift register generator. Its initial bit values are used (1111).

The output sequence Z_n : 011111000000001 Generated by NLFFSR in is periodic of period 15, which is the same as the period of the sequence generated by NLFFSR of 4 bits.

Period of the sequence generated by NLFFSR is maximum if we use the primitive polynomial. To design any stream cipher system, one needs to consider the NLFFSR with primitive feedback polynomials as the basic building blocks. Period of the enciphering sequence can be increased if it is generated by following methods:

1. Addition of maximal length sequences.
2. Multiplication of maximal length sequences.
3. Using multi logic generalized linear feedback shift register.

The usefulness of these sequences depends in large part on there having nearly randomness properties. Therefore such sequences are termed as pseudorandom binary sequences. The balance, run and correlation properties of these sequences make them more useful in the selection of secret keys[17-20]. The NLFFSR generated sequences are of great importance in many fields of engineering and sciences.

Steganography using T-codes, Turbo codes and NLFFSR: The design principle of Steganographic systems is based on the premise that most communication channels-such as telephone lines and radio broadcasts-transmit signals accompanied by some kind of noise. This noise can be replaced by a transformed secret signal indistinguishable from the noise without the secret key. In this way, the secret signal can be transmitted undetected. Hiding data in an image requires two files. The first file, called the "cover-image", will be the innocent looking image that holds the hidden information. The second file will contain the information to be hidden. When combined, the two files will generate a "stego-image". There exists a number of ways to create a stego-image from the given cover-image and the data to be hidden. These include least significant bit (LSB) insertion, masking and filtering, redundant pattern encoding, LSFR insertion and other spread spectrum methods. The insertion techniques using LFSR/NLFFSR (Pseudo random sequence) are much stringer data hiding techniques than LSB insertion methods. The NLFFSR (Non Linear Feed Forward Shift Register) insertion technique is used in this study .The procedure is clearly illustrated by Fig. 8.

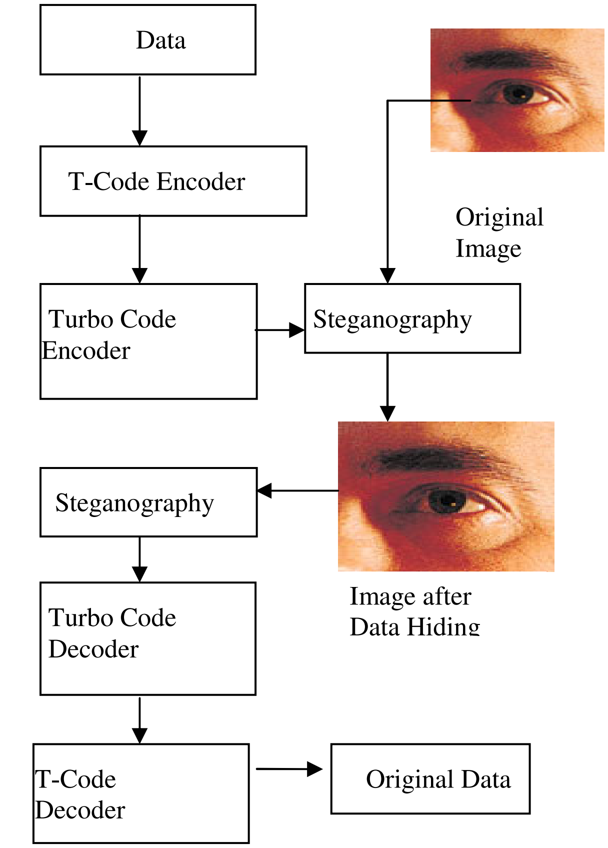

Fig. 8: The block diagram of the proposed security system

The data stream is first encoded using T-Codes, after Turbo Code encoder processes that data. The data stream after processing becomes robust and secure for hiding in Stego-Object, which is a digital image in this case. The Steganography tool hides the processed data stream in Image using NLFFSR substitution method. The preprocessing of data before hiding enables us to trust even weaker methods of Steganography if employed by the Steganography tool. After then, at the receiver end, reverse procedure is applied to recover the data stream.

Let us take a example, A 640*480 image, which has the color depth of 24 bits, has the potential to hide a total of 921600 bits of information. Suppose the original raster data in 8 pixels of a 24-bit image is:

00100111 11101001 11001000
00100111 11001000 11101001
11001000 00100111 11101001
00100111 11101001 11001000
00100111 11001000 11101001
11001000 00100111 11101001
00100111 11101001 11001000
00100111 11001000 11101001

To hide the letter ‘A’ in this image we take its binary value (after T-code and Turbo Coding): 00101000

(a) (b)

Fig. 9: (a) Original image, (b) Stego image

This value is inserted in the image by the pseudo random binary sequence generated by NLFFSR taken for the data hiding process, the binary sequences are taken in the pair of two which gives the decimal value ranging from 0 to 3 which means we consider only the last four bits of each pixel for modification. The nature of the pseudo random binary sequence ensures that we get equally random decimal sequence corresponding to which we will hide the data stream in the image. Suppose we get the decimal sequence as 2, 0, 1, 2, 3, 2, 1 and 0…

After hiding the data according to this sequence we get:

00100111 11101001 11001**0**00
00100111 11001000 1110100**0**
11001000 00100111 111010**1**1
00100111 11101001 11001**0**00
00100111 11001000 1110**1**001
11001000 00100111 11101**0**01
00100111 11101001 110010**0**0
00100111 11001000 1110100**0**

The underlined bold bits are the only bits that actually changed from the original raster 192-bit data. The bold bits are the data, which is hidden but not modified. To an average human eye, this slight change in the original image is indistinguishable from the previous image. This small ratio of the modified bits makes it very difficult for a person to observe any changes and hence conclude that any secret information is hidden. Therefore, we will be able to send a hidden ‘A’ undetected. Many trials of this process show that the use of NLFFSR in the process of hiding the data in the image yields very good results in terms of bit changes. Also the randomness in selecting the bit for the modification provides better security and protection. It is a major improvement in this technique. In fact, we can hide the data in the least and four least significant bits and still the human eye will not be able to see the difference.

Table 1: For comparisons

Attacks	Image PSNR(dB) Under attacks	Extracted watermark PSNR(dB)
2X4 Median Filtering	26	18
4X4 Median Filtering	24	28
Gaussian Filtering 3X3	31	22
JPEG Compression of 50%	32	31
JPEG Compression of 50%	31	23
JPEG Compression of 50%	29	22
Sharpening 3X3	19	54

EXPERIMENTAL RESULTS

The experimental results are shown in the Fig. 9. Comparisons are shown in the Table 1 we have performed various types of the attacks on the stego-image to check its robustness. We are also calculating the PSNR for the performance of the data.

CONCLUSION

In this study, we have presented a review of the field of Steganography and concentrated on improving the security of the data transferred by first encoding the data using T-Codes and thereafter using Turbo Codes. When T-codes are used to encode an information source most efficiently, their tendency towards self-synchronization is among the strongest. The best T-codes achieve self-synchronization within 1.5 characters following a lock loss. Using T-Codes help us synchronize data stream as well compress it, also providing it added invisibility if active intruders manage to extract the data from the Stego-Object. Turbo-codes have attracted a great deal of interest since their discovery in 1993, due to their ultimate limits of the capacity of a communication channel. Using Turbo Codes makes our technique robust against channel errors. As illustrated the technique of Turbo Coding enables us recover block errors, which is very beneficial in technique like Steganography, in which the media object may undergo some changes while in transit. We use the data hiding technique-using pseudo random sequence, which make it more complex. This technique is best when we need to send data with very high security.

REFERENCES

1. Bender, W., D. Gruhl, N. Morimoto and A. Lu, 1996. Techniques for data hiding. IBM Systems J., 35: 313-336.
2. Johnson, N.F. and S. Jajodia, 1998. Exploring steganography: Seeing the unseen. IEEE Computer, 31: 26-34.
3. Anderson, R.J. and F.A.P. Petitcolas, 1998. On the limits of steganography. IEEE J. Selected Areas in Commun., 16: 4.
4. Titchener, M.R., 1984. Digital encoding by means of new T-codes to provide improved data synchronization and message integrity. IEE Proc., Comput. Digit. Tech., 131: 151-153.
5. Higgie, G.R., 1996. Database of best T-codes. IEE Proc. E. Comput. Digit. Tech., 143: 213-218.
6. Titchener, M.R. and J.J. Hunter, 1986. Synchronization process for the variable-length T-codes. IEE Proc. E. Comput. Digit. Tech., 133: 54-64.
7. Berrou, C., A. Glavieux and P. Thitimajshima, 1993. Near Shannon limit error-correcting coding: Turbo codes. Proc. IEEE Intl. Conf. Commun., Geneva, Switzerland, pp:1064-1070.
8. Schurgers, C., L. Van der Perre, M. Engels and H. de Man, 1998. Adaptive turbo decoding for indoor wireless communication. Proc. ISSSE'98, URSI Int. Symp. Signals, Systems and Electronics, pp: 107-111.
9. Lin, S. and D.J. Costello, 1983. Error Control Coding: Fundamentals and Applications. Prentice-Hall.
10. Ten Brink, S., 1999. Convergence of iterative decoding. Electron. Lett., 35: 806-808.
11. Anil Kumar and Navin Rajpal, 2004. Turbo codes for error control code. J. Inform. Technol., 2: 29-33.
12. Nikolaidis, N. and I. Pitas, 1998. Robust image watermarking in the spatial domain. Signal Process, 66: 385-403.
13. Bruyndonckx, O., J.J. Quisquater and B. Macq, 1995. Spatial method for copyright labeling of digital images. Proc. IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, Greece, 20-22 Jun, pp: 456-459.
14. Jiwu Huang, Yun Q. Shi and Yi Shi, 2000. Embedding image watermarks in DC components. IEEE Trans. CSVT, 10: 974-979.
15. Shinfeng D. Lin and Chin-Feng Chen, 2000. A robust DCT-based watermarking for copyright protection. IEEE Trans. Consumer Electron., 46: 415-421.
16. Shannon, C.E., 1948. A mathematical theory of communication. Bell Syst. Tech. J., Jul. and Oct.1948, 27: 379-423 and 623-656. http://galaxy.ucsd.edu/new/external/shannon.pdf
17. Rajpal, N., A. Kumar, S. Dudhani and P.R. Jindal, 2004. Copyright protection using non linear forward feedback shift register and error correction technique. 7th Ann. Intl. Conf. Map India, New Delhi, India.
18. Rajpal, N., A. Kumar and P.R. Jindal, 2004. Demonstrating the use of error coding technique in the field of steganography, along with linear feedback shift register technique. 2nd Workshop on Computer Vision, Graphics and Image Processing, Gwalior, India, pp: 22-27.
19. Rajpal, N., A. Kumar, P.R. Jindal and A. Saroagi, 2004. An investigation into the use of linear feedback shift register for data encrypting and data hiding in the field of steganography. Conf. e-security, Cyber Crime and Law. Chandigarh, India.
20. Ahmad, A., M.J. Al-Musharafi, S. Al-Busaidi, A. Al-Naamany and J.A. Jervase, 2001. An NLFSR based sequence generation for stream ciphers. Proc. Intl. Conf. Sequences and their Applications (SETA '01), pp: 11-12.