Steganography Using Error Correction Methods
الإخفاء باستخدام طرائق اكتشاف الخطأ

د. وسام سمير بهية/ جامعة بابل/ قسم علوم الحاسبات
صفاء مهادي عبيس/ جامعة بابل/ قسم علوم الحاسبات

1. Abstract

In this research, we benefit from error correction methods to hide some information. By hiding some bits in sent data, i.e. making benefit error, and then using error detection methods to get hidden bits and return original data. Here, we use Hamming code error detection method to apply this idea.

[illegible] Hamming Code [illegible].

2. Introduction

Networks must be able to transfer data from one device to another with complete accuracy. A system that cannot guarantee that the data received by one device are identical to the data transmitted by another device is essentially useless. Reliable systems must be have a mechanism for detecting and correcting the errors and this is done by using error detection and correction methods (Welch J. C., 2000).

Steganography and cryptography are cousins in the spy craft family. Cryptography scrambles a message so it can not be understood. Steganography hides the message so it can not be seen (Anderson R. J. and Petitcolas F. A., 1998).

In this research, we combine one of error correction techniques (that is hamming code) with pure steganography in the process of sending a data between two computers over a network**.**

3. Steganography

Information hiding is the art of hiding message inside other message. In other words, Information hiding is studies the ways that make communication invisible by hiding secret in innocuous message. This part of information hiding called steganography. Steganographic methods usually hide messages in other, harmless looking data in such that third person can not detect or even prove this processes (Katzenbeisser S.& Petitcolas F., 2000).

There are three main types of steganography which are:

- **Pure Steganography**

We call a steganographic system which does not require the prior exchange of some secret information (like a stego-key) pure steganography (Kahn, D. 1996).

- **Secret Key Steganography**

  A secret key steganography system is similar to a symmetric cipher: the sender chooses cover C and embeds the secret message into C using a secret key K. if the key used in the embedding process is known to the receiver, he can reverse the process and extract the secret message. Anyone who does not known the secret key should not be able to obtain evidence of the encoded information (Trivedi S. and Chandramouli R., 2005).

- **Public Key Steganography**

  Public key steganography system require the use of two keys, one is private and the other one is public key; the public key is stored in a public database whereas the public key is used in the embedding process, the secret key is used to reconstruct the secret message (Backes M. and Cachin C., 2005).

From the types of steganography above, our research is pure steganography.

4. Error Correction

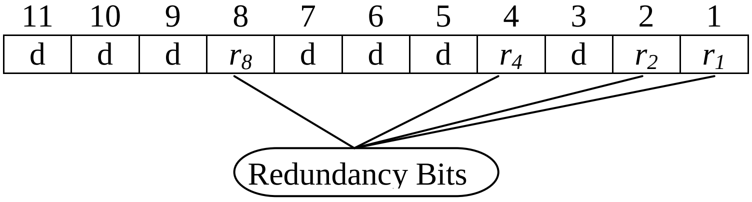
Error correction can be handled in two ways. In one, when an error is discovered, the receiver can have the sender retransmit the entire data unit. In the other, a receiver can use an error correcting code, which automatically corrects certain errors. In theory, it is possible to correct any binary code errors automatically. Error correcting codes however are more sophisticated than error detection codes and require more redundancy bits. The number of bits required correcting a multiple-bit or burst error is so high that in most cases it is inefficient to do so. For this reason, most error correction is limited to one-, two-, or three-bit errors. The error correction method that is used in this research is single bit error correction.

To correct the error, the receiver simply reverses the value of the altered bit. To do so, however, it must know which bit is in error. The secret of error correction, therefore, is to locate the invalid bit or bits and this is done by using redundancy bits. To calculate the number of redundancy bits (*R*) required to correcting a given number of data bits (*M*), there is a relationship between (*M*) and (*R*) that is (Forouzan B., 1998) :

$$2^R \geq M + R + 1$$

4.1 Positioning the Redundancy Bits

The Hamming code (Hamming R., 1950) can be applied to data units of any length and uses the relationship between data and redundancy bits shown above. For example, a seven-bit ASCII code requires four redundancy bits that can be added to the end of the data unit or interspersed with the original data bits. In figure (1), these bits are placed in positions 1, 2, 4, and 8 (the positions in an 11-bit sequence that are powers of 2). For clarity in the examples below, we refer to these bits as *r_1, r_2, r_4, and r_8.*



11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

Figure (1), Locations of Redundancy Bits

The XORing combinations used to calculate each of the four *r* values for a seven-bit data sequence are as follows (Forouzan B., 1998) :

r_1 bits: 1, 3, 5, 7, 9, 11

r_2 bits: 2, 3, 6, 7, 10, 11

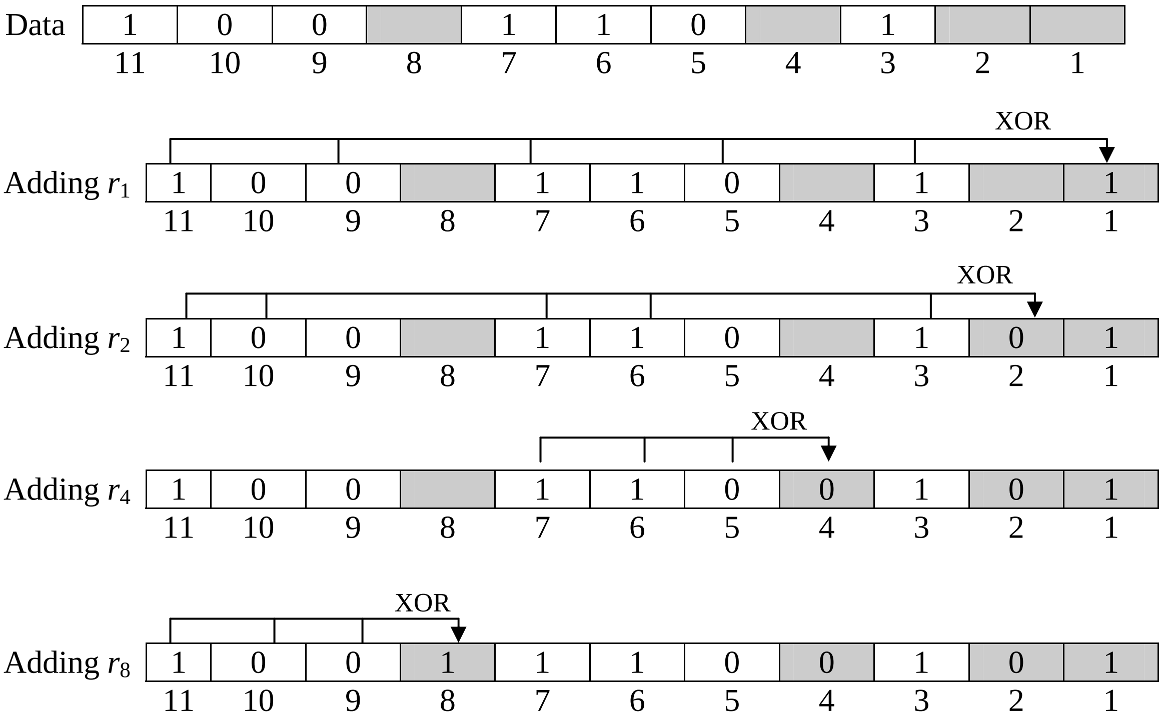
r_4 bits: 4, 5, 6, 7

r_8 bits: 8, 9, 10, 11

This is according to Hamming code method (even parties), for more information see (Forouzan B., 1998) .

4.2 Calculating the r Values

Figure (2) shows a Hamming code implementation for an ASCII character. In the first step, we fill in each bit of the original character in its appropriate position in the 11-bit unit. In the subsequent steps, we calculate even parities for the various bit combinations. The parity value for each combination is the value of the corresponding *r* bit. For example, the value of r_1 is calculated to provide even parity for a combination of bits 3, 5, 7, 9, and 11. The value of r_2 is calculated to provide even parity with bits 3, 6, 7, 10, and 11. The final 11-bit code is sent through the transmission line (Forouzan B., 1998) .



	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0		1	1	0		1		
Adding r_1	1	0	0		1	1	0		1		1
Adding r_2	1	0	0		1	1	0		1	0	1
Adding r_4	1	0	0		1	1	0	0	1	0	1
Adding r_8	1	0	0	1	1	1	0	0	1	0	1

Code: 1 0 0 1 1 1 0 0 1 0 1

Figure (2),Original Data with Redundancy Bits

4.3 Error Detection and Correction

Now imagine that by the time the above transmission is received, the 7th bit has been changed from 1 to 0 (see figure 3).

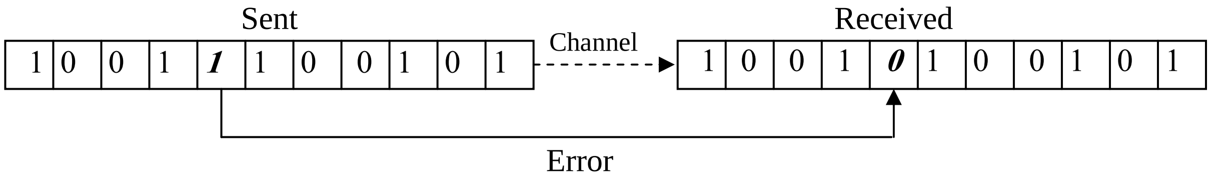


Figure (3), Received Data With Error

The receiver takes the transmission and recalculates four new values using the same sets of bits used by the sender plus the relevant parity *(r)* bit for each set (see figure 4). Then it assembles the new parity values into a binary number in order of *(r)* position (r_8, r_4, r_2, r_1). In our example, this step gives us the binary number 0 1 1 1 (7 in decimal), which is the precise location of the bit in error. Once the bit is identified, the receiver can reverse its value and correct the error(Forouzan B., 1998) .

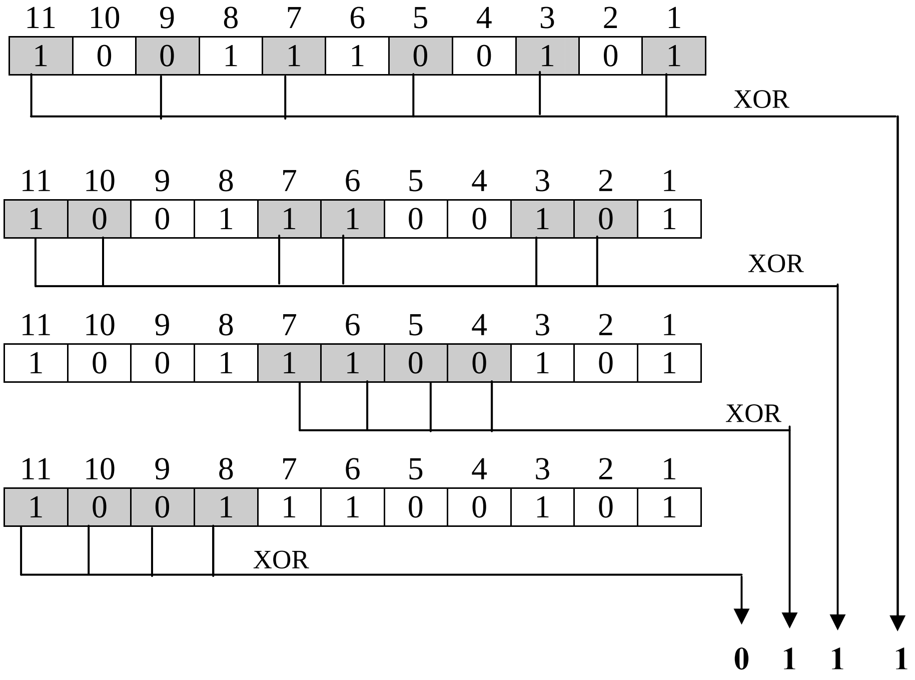


Figure (4), Finding Position of Error

5. Proposed Method

The classical hiding system hide image in audio or text in image or some combination from them. The aim from our research is to hide information (text or image) in the transmitted data between two computers over network. When data transmit, it must segmented to blocks (each block is 7-bit) and the error correction information is computed for each block. So each block become 11-bit (after adding redundancy bits) after that each block sent to the receiver. The receiver check if an error happens in the received data and removes the redundancy bits. But what happens when we intend an error in the senting data? The error we mean is the hidden data, so the receiver when find the error, he retains a copy from error and correct it. The collection of errors is the hidden message. figure (5) shows the proposed method. The hidden bit location is selected randomly.

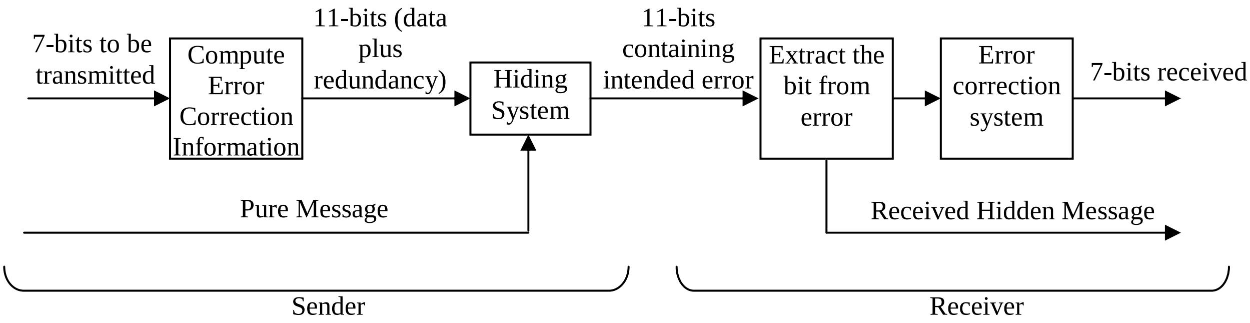

7-bits to be transmitted
Compute Error Correction Information
11-bits (data plus redundancy)
Hiding System
11-bits containing intended error
Extract the bit from error
Error correction system
7-bits received
Pure Message
Received Hidden Message
Sender
Receiver

Figure (5), The Proposed Hiding System

Example:

Assume the block of data is (1 0 0 1 1 0 0) and we want to hide bit value (1). We Firstly compute error correction information as in the section 4.2:

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	1	1	0	0	0	1	0

After computing error correction information, the 7-bits become 11-bits (data with redundancy). Now we want to hide the bit value (1), so we select a location randomly with one condition that is the value of selected location must be the reverse of the bit value to be hiding. The location (5) is suitable so we change the value of it to (1) as in figure (6).

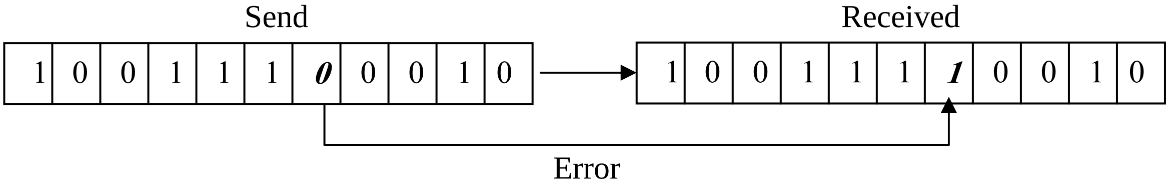


Figure (6), Hide a bit 1 in the 5th Location of Sent Data

The receiver then check the received data by computing the correction information again as previously stated in figure (4). The resulting value is (0 1 0 1) which means (5) in decimal, so the error in location (5) that is the hidden bit. The receiver gets the value of fifth bit as the transmitted hidden bit and reverses the fifth bit value from (1) to (0), in order to return the original message.

6. Discussion

It can be noted the following points from the suggested method:

1. The proposed method used intended error to hide the data without any influence on it.
2. There is no need to specify the location of hidden bit because the receiver can detect it, thus it is hard to detect.
3. The value of hidden bit must be the opposite to the original data bit (i.e. we must hide bit 0 in bit 1, or vise versa, to ensure make error in the original data).
4. Extra errors may also produce accepted results to steganography.
5. We can use the proposed method with three error detection technique, for example, changing one as hidden bit and leave others for additional noise errors.

7. References

Anderson R. J. and Petitcolas F. A., (May 1998**) "On the limits of steganography",** IEEE Journal on Selected Areas in Communications, vol. 16.

Backes M. and Cachin C., (2005) **"Public-key steganography with active attacks",** in Proc. 2nd Theory of Cryptography Conference (TCC 2005) (J. Kilian, ed.), vol. 3378 of Lecture Notes in Computer Science, pp. 210–226, Springer.

Forouzan B., (1998) "***Introduction to Data Communications and Networking***", McGraw-Hill Companies.

Hamming R. W., (1950), **"Error Detecting and Error Correcting Codes"**, Bell System Technical Journal, vol. 29, pp. 147-160.

Kahn, D. (1996) **"Codebreakers: The Story of Secret Writing"**, Revised ed., Scribner, New York.

Katzenbeisser S. and Petitcolas F., (2000) "***Information Hiding Techniques for Steganography and Digital Watermarking***", Artech House Inc.

Trivedi S. and Chandramouli R., (Feb. 2005) **"Secret key estimation in sequential steganography",** Supplement on Secure Media, IEEE Trans. on Signal Processing.

Welch J. C., (2000) **"The Technical Introduction To Networks"**, The Journal of Macintosh Technology, Volume Number: 16, Issue Number: 4.