International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 20 Issue 2 – FEBRUARY 2016.

AN APPROACH FOR LOSSLESS DATA HIDING USING LZW CODES

Sindhuja J[1],Anitha B[2]

[1]*PG Scholar, Department of Information Technology, Kongu Engineering College, Perundurai,Erode-638 052,Tamil Nadu,India.*
[2]*Assistant Professor, Department of Information Technology, Kongu Engineering College, Perundurai,Erode-638 052,Tamil Nadu,India.*

***Abstract*--Hiding a message in compression codes can reduce transmission costs and simultaneously make the transmission more secure. In high-performance, data-hiding Lempel–Ziv–Welch (HPDH-LZW) scheme, which reversibly embeds data in LZW compression codes by modifying the value of the compression codes, where the value of the LZW code either remains unchanged or is changed to the original value of the LZW code plus the LZW dictionary size according to the data to be embedded. Compared to other information-hiding schemes based on LZW compression codes, the proposed scheme achieves better hiding capacity by increasing the number of symbols available to hide secrets and also achieves faster hiding and extracting speeds due to the lower computation requirements.**

Keywords
LZW, Steganography, Information hiding.

I. INTRODUCTION

With the rapid development of new Internet techniques, huge amounts of data are generated on the Internet daily. With the extensive, worldwide use of the Internet, it is now necessary to encrypt sensitive data before transmission to protect those data. Reversible data-hiding techniques can ensure that the receiver can receive hidden messages and recover needed data without distortion. Reversible data-hiding has received extensive attention since recoverable media are more useful when protecting the security and privacy of sensitive information. For example, assume that the personal information of a patient is private information and the patient's X-ray images are used as cover media. It is very important to recover the X-ray images without any loss of detail after retrieving the patient's personal information. Currently, reversible data-hiding schemes are applied in three domains, i.e., the spatial domain, the transformed domain and the compression domain. In the spatial domain, the values of the pixels of the cover image are altered directly to hide the data. In the transformed domain, the cover image is processed by a transform algorithm to obtain the frequency coefficients. Then, the frequency coefficients are modified to hide the data. In the compression domain, the compression code is altered to hide the data. LZW coding is a simple,well-known, lossless compression algorithm that compresses and decompresses data by using a dictionary that is automatically produced, so LZW coding eliminates the need to analyze the source file or transmit any auxiliary information to the decoder.

The related DH-LZW scheme based on the LZW algorithm hides the data by shrinking one character of one symbol to hide the data. However, the hiding capacity was low because only the symbol whose length is greater than the threshold can hide secret data and an embeddable symbol hides only one secret bit.

The HCDH-LZW scheme is used to improve the performance of Shim, Ahn, and Jeon's method by shrinking the characters according to the length of the symbol used to hide the data, thereby achieving higher embedding capacity. This hiding capacity is higher because more symbols are available to hide secret bits and because one symbol can hide more than one secret bit. However, only symbols with lengths larger than the threshold can hide data and repeated symbols increase the size of the dictionary, which, in turn, lowers the hiding speed. In addition, the extracting algorithm is very complicated, and this increases the computation costs. Further, both scheme must transmit auxiliary information, the threshold value.

To overcome the shortcomings of these methods, the proposed data-hiding scheme that is based on LZW codes by utilizing the relationship between the output compression codes and the size of the dictionary. The proposed scheme guarantees that the receiver can recover the source data and extract the hidden data without loss. In comparison to other proposed schemes, our scheme can achieve a much higher embedding capacity and lower computation costs.

The need for data hiding is such that the existence of the message is not known to anyone apart from the sender and the intended receiver. In data hiding, the receiver can able to recover only the hidden data and not the source data which is used as a cover medium.

Cover Medium

This is the medium in which he/she wants to hide data. It can be an image or text data or any audio or video file. It is also called as source data.

Classification of cover medium

In modern approach, depending on the nature of cover object, it can be divided as follows,

Text cover medium

Hiding information in plain text can be done in many different ways. Many techniques involve the modification of the layout of a text, rules like using every n^{th} character or the altering of the amount of white space after lines or between words.

Image cover medium

Image steganography is steganography technique using image as cover. It uses the fact that human visual system is having low sensitivity to small changes in digital data. It modifies pixel values of image for data hiding.

Audio cover medium

In audio steganography system, the cover medium is digital Audio. Secret messages are embedded in digital sound. Some common methods used in audio steganography are LSB coding, parity coding, phase coding, spread spectrum and echo hiding.

Video cover medium

Video files are generally a collection of images and sounds, so most of the presented techniques on images and audio can be applied to video files too. The great advantages of video are the large amount of data that can be hidden inside and the fact that it is a moving stream of images and sounds.

The reversible data hiding allows the receiver to recover the source data and extract the hidden data without any loss. Reversible data hiding schemes can be applied in three domains.

In the spatial domain, the values of the pixels of the cover image are altered directly to hide the data. In the transformed domain, the cover image is processed by a transform algorithm to obtain the frequency coefficients and then the frequency coefficients are modified to hide the data.

In the compression domain the compression code is altered to hide the data. The merit of using reversible data hiding schemes in the compression domain is that such schemes can reduce transmission costs and simultaneously secure the information that is transmitted.For example, assume that the personal information of a patient is private information and the patient's x-ray images are used as cover media. The receiver should be able to recover both the x-ray image and personal information of a patient which is hided in the x-ray image without any loss. This is an example of reversible data hiding technique in medical field.

The reversible data hiding techniques are used in various applications such as military, science and education, digital image processing and in various domains. It is used to recover both the source data and the hidden data.

Compression is a reduction in the number of bits needed to represent data. Compressing data can save storage capacity, speed file transfer, and decrease costs for storage hardware and network bandwidth. The term data compression refers to the process of reducing the amount of data required to represent a given quantity of information. The aim of data compression is to reduce redundancy in stored or communicated data which increases data density. It can be applied in file storage, distributed systems and data transmission.

The main idea of data compression is to reduce the quantity or amount of data to be sent or transmitted. Various algorithms are used for data compression technique to reduce the size of the file to be transmitted without the degradation of the original file. There are different types of data compression present in compression technique which can be used for compression of text.

Compressing data can be a lossless or lossy process. Lossless compression enables the restoration of a file to its original state, without the loss of a single bit of data, when the file is uncompressed. Lossless compression is the typical approach with executables, as well as text and spreadsheet files, where the loss of words or numbers would change the information. A simple characterization of data compression is that it involves transforming a string of characters in some representation into a new string which contains the same information but whose length is as small as possible

.Lossy compression permanently eliminates bits of data that are redundant, unimportant or imperceptible. Lossy compression is useful with graphics, audio, video and images, where the removal of some data bits has little or no discernible effect on the representation of the content. Graphics image compression can be lossy or lossless.

The main advantages of compression are a reduction in storage hardware, data transmission time and communication bandwidth, and the resulting cost savings.A compressed file requires less storage capacity than an uncompressed file, and the use of compression can lead to a significant decrease in expenses for disk and drives. A compressed file also requires less time for transfer, and it consumes less network bandwidth than an uncompressed file. Many data processing applications require storage of large volumes of data and no of such applications are increasing constantly as the use of computers.

II. Related works

In this section, the LZW compression, DH-LZW and HCDH-LZW schemes are described briefly.

2.1. The LZW algorithm

The LZW algorithm compresses the source file mainly by substituting fixed length codes for the variable length sequential symbols of the source file. Since the dictionary initializes with ASCII values from 0 to 255, the shortest code length is nine bits. The encoder reads the source data sequentially, and if the sequences are in the dictionary, the code that corresponds to the previous symbol that was scanned will be the output; otherwise, the sequences are placed into the next unused dictionary location. The longer the symbols in the dictionary are, the higher the LZW compression ratio is. Since the decoder constructs the same dictionary dynamically and automatically in the decoding phase, the sender does not have to send the entire dictionary to the receiver. Then, the decoder recovers the source file by converting the LZW codes to the corresponding symbols according to the dictionary.

2.2. The DH-LZW scheme

The main idea was to shrink one character of one symbol to hide the data. Shim, Ahn, and Jeon's scheme sets THD as a threshold to decide whether or not a symbol can be used to hide data. During the hiding phase, a symbol can be used to hide one secret bit only when the length of the symbol is larger than THD. The hiding strategy must match the parity of the symbol's length to the secret bit by shrinking the last

character of the symbol. The parity bit of the odd number is 1, and the parity bit of the even number is 0.

So if the secret bit equals the parity bit of the symbol's length, then the symbol does not shrink; otherwise, the symbol shrinks the last character, and the shrunken character is returned to the source file. Since the symbol's length increases gradually during the construction of a dictionary, the shrunken symbol already exists in the dictionary. The extracting phase just tests the parity bit of the symbol. There may be a secret bit when a new symbol is added to the dictionary.

If the symbol's length is larger than THD, the secret bit equals the parity bit of the previous symbol's length. If the symbol's length is equal to THD, there are two possibilities. One possibility is that the symbol has existed in the dictionary, which means that the symbol hides one secret bit that is equal to the parity bit of the previous symbol's length else the symbol does not hide the data.

2.3. The HCDH-LZW scheme

The main idea of the HCDH-LZW scheme is to shrink the symbol until it is as short as possible while hiding as much data as possible. If current symbol's length is greater than 2, it can be used to hide data. The larger the symbol's length is, the more data it can hide. This scheme still uses the LZW code as output. When the current symbol's length is greater than 2, the encoder can hide secret bits by shrinking the symbol. When the symbol's length equals 2, the previous symbol's length equals 1, and the symbol cannot be shrunk. The number of secret bits that an embeddable symbol can hide is the logarithm of its previous symbol's length.

For example, if the previous symbol's length is 2 or 3, it can be used to hide 1 bit, and if the previous symbol's length is between 4 and 7, it can be used to hide 2 bits. The main idea of the extracting phase is to examine the following decoded characters to count the secret bits. For example, if the symbol combines with two characters of the following symbol still in the dictionary, then the hidden secret bits are "10". In the extracting phase, if the secret value is 0, then the symbol remains unchanged; otherwise, it is shrunk according to the secret value.

III. THE PROPOSED WORK

The main idea of the proposed scheme is to modify the value of the LZW codes to hide a secret rather than modifying the content of the dictionary. Every embeddable symbol can be used to hide one secret bit. Before a new symbol is added to the dictionary, the encoder modifies the value of the output LZW code according to the secret value. If the secret bit is '0', the output is the original LZW code and if the secret bit is '1', the output is the sum of the value of the LZW code and the current size of the dictionary. The hiding algorithm is summarized as follows,

Input Source file and secret file.
Output LZW codes.
Step 1 Get the first character c_0 from the source file.
Set $s = c_0$, where s is a string variable.
Step 2 If s exists in the dictionary,
Set the previous symbol $s_p = s$.
Get the next character c from the source file and set s = s||c,
where || means the concatenation operation.
Else
Get next secret bit b.
Get the code C, where C is the dictionary indices of s_p .
Step 3 If b=1,set C=C+size, where size is the the current size of the dictionary.
Step 4 Ouput c and add s into thedictionary
Set $S=c_s$, where c_s is the last character of s.

In this scheme, once a new symbol is added into the dictionary, that symbol can be used to hide a secret. As a result, the number of bits for hiding secret information is equal to the number of new symbols. The data hiding phase modifies the value of the LZW compression code to hide secrets, except for the initial 256 symbols in the dictionary.

In the data extracting process, assume that the value of the current processing code is C and the current size of the dictionary is Size. If C is larger than size, then he extracted secret bit is '1',otherwise, the secret bit is '0'. And if the extracted secret bit is '1',then the original LZW code is the difference between C and Size. If the extracted secret is '0',then the original LZW code is C. The data extracting algorithm is summarized as follows,

Input LZW Codes
Output Source file and secret file
Step1: Get a LZW code $\mathbf{C_0'}$.
Step2: If $\mathbf{C_0'} > \mathbf{Size'}$, where $\mathbf{Size'}$ is the current size of the dictionary, set $\mathbf{C_0' = C_0' - Size'}$
Step3: Set secret bit = '1'.
Step4: If $\mathbf{C_0' < Size'}$, Set secret bit = '0'.
Step5: Output $\mathbf{S'}$, where $\mathbf{S'}$ is the symbol of $\mathbf{C_0'}$ in the dictionary.
Step6: Get character $\mathbf{C_s'}$, where $\mathbf{C_s'}$ is the first character of $\mathbf{S'}$.
Step7: Set $\mathbf{O' = C_0'}$, where $\mathbf{O'}$ is the old code.
Step8: Get the next LZW code $\mathbf{C'}$.
Step9: If $\mathbf{C_0' > Size' + 1}$, set $\mathbf{C' = C' - Size' - 1}$ and set secret bit = '1'.
Step10: Otherwise set secret bit = '0'.
Step11: If $\mathbf{C'}$ is not in the dictionary, set $\mathbf{S' = S_0' \| C_s'}$. $\mathbf{S_0'}$ = string variable.
Step12: Otherwise $\mathbf{S' = S_0'}$, where $\mathbf{S_0'}$ is the symbol of $\mathbf{O'}$ in the dictionary and output $\mathbf{S'}$
Step13: Set $\mathbf{S_{new}' = S_0' \| C_s'}$, where $\mathbf{S_{new}'}$ is a new symbol.
Step14: Add $\mathbf{S_{new}'}$ into the dictionary.
Step15: Set $\mathbf{O' = C'}$ and continue to step 8.

In the example, the source file is "sddsddssddsdsddsddsd," and the secret file is "1001000100."In the following table, the first secret bit is '1', and since 256 symbols existed in the dictionary before the data hiding procedure, the output code is the value of the original code plus the current size of the dictionary, i.e., 371.

After generating the 257th item, since the secret bit is '0', the second output code remains unchanged as 100. In

the extracting phase, the first LZW code is 371, which is larger than the current dictionary size of 256, so the hiding secret bit is '1', and the extracted symbol is "s". The second LZW code is 100, which is smaller than the current dictionary size 257, so the secret bit is '0', and the extracted symbol is "d".

Data Hiding

Input	Originalcode	Output	New item	Hidden bit
sd	115	371	256=sd	1
d	100	100	257=dd	0
s	100	100	258=ds	0
dd	256	515	259=sdd	1
ss	258	258	260=dss	0
dds	259	259	261=sdds	0
ds	256	256	262=sds	0
ddsd	261	524	263=sddsd	1
dsd	257	257	264=dds	0
d	256	256		0

Data Extracting

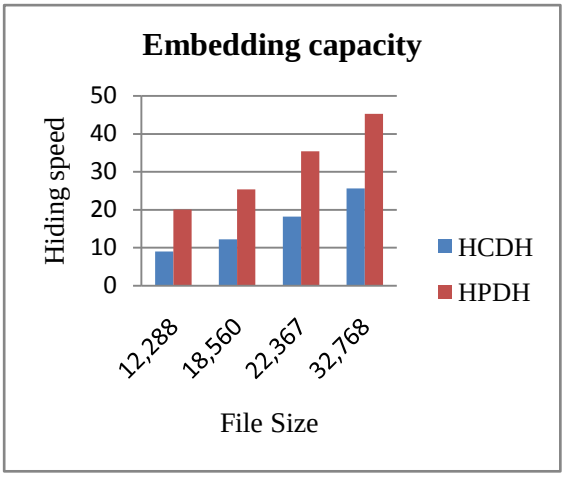
Input	Output	New item	Extracted bit
371	s		1
100	d	256=sd	0
100	d	257=dd	0
515	sd	258=ds	1
258	ds	259=sdd	0
259	sdd	260=dss	0
256	sd	261=sdds	0
524	sdds	262=sds	1
257	dd	263=sddsd	0
256	sd	264=dds	0

The proposed scheme increases the embedding capacity by increasing the number of embeddable symbols. The increased hiding and extracting speeds of the proposed scheme are the result of the simple computation of the proposed scheme. Moreover, the proposed scheme decreases the dictionary's size because there is no modification of the content of the dictionary during the data hiding phase.

This schemeis based on the LZW compression code but modifies the value of the LZW compression codes to embed secret data. The proposed scheme increases the number of symbols available to hide secrets and does not change the content of the dictionary. Since the maximum number of hidden bits in the proposed scheme is equal to the size of the dictionary, it achieves much higher embedding capacity than HCDH-LZW. In addition, the proposed scheme achieves faster hiding and extracting speeds than HCDH-LZW. Also, the dictionary generated by our proposed scheme is much smaller than that for HCDH-LZW.

IV. PERFORMANCE ANALYSIS

Four text files are taken to analyze the performance of the proposed system. The size of each file is measured in bytes.



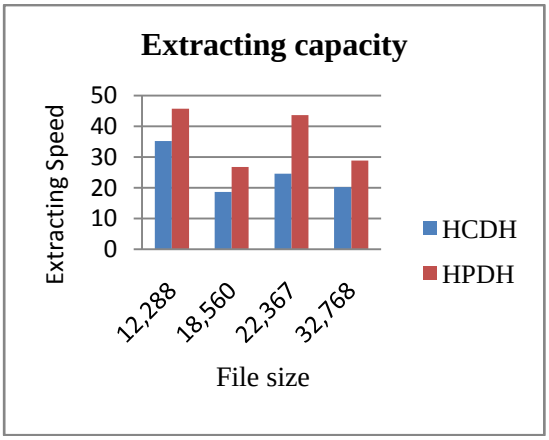


Figure 6.1 Embedding and Extracting capacity

In Figure 6.1 Embedding capacity graph, the comparison between the existing system and the proposed system is shown. The embedding capacity is increased according to the file size. Thus the data hiding speed increases in high performance lossless data hiding scheme.

V. CONCLUSION AND FUTURE WORK

In the proposed scheme, the value of the LZW compression code is modified to embed the secret data. The proposed scheme increases the number of symbols available to hide secrets and does not change the content of the dictionary. It achieves high embedding capacity and faster hiding and extracting speed than HCDH-LZW. The dictionary generated is also much smaller than the HCDH-LZW. From the results it can be observed
that the proposed scheme works better when compared to the existing system and high embedding capacity is

achieved. This scheme can be applied with efficient version of LZW algorithm which can be taken as the future work.

REFERENCES

[1] Chang C.C, Lee C.F, Chuang L.Y, (2009), 'Embedding secret binary message using locally adaptive data compression coding', International Journal of Computer Sciences and Engineering Systems, Vol.3, No.1, pp 55-61.

[2] Chang C.C, Lin C.Y, (2007), 'Reversible Steganographic method using SMVQ Approach based on declustering', Information Sciences, Vol.177, No.8, pp 1796-1805.

[3] Chang C.C, Lu T.C, (2006), 'Reversible index domain information hiding scheme based on side-match vector quantization', Journal of Systems and Software, Vol.79, No.8, pp.1120-1129.

[4] Chang C.C, Wu W.C, (2006), 'A Steganographic method for hiding secret data using side match vector quantization', IEICE Transactions on Information and Systems, Vol.8, No.9, pp.2159-2167.

[5] Chen C.C, Chang C.C, (2010), 'High Capacity reversible data hiding for LZW codes', In proceedings of the second International conference on Computer Modeling and Simulation, No.1, pp.3-8.

[6] Chen W.J, Huang W.T, (2009), 'VQ indexes compression and information hiding using hybrid lossless index coding', Digital Signal Processing, Vol.19, No.3, pp.433-443.

[7] Jo M, Kim H.D, (2002), 'A Digital image watermarking scheme based on vector quantization', IEICE Transactions on Information and Systems, Vol.85, No.6, pp.1054-1056.

[8] Lu Z.M, Wang J.X, Liu B.B, (2011), 'An improved lossless data hiding scheme based on image VQ index residual coding', Journal of Systems and Software, Vol.82, No.6, pp.1016-1024.

[9] Ma K, Zhang X, Yu N, Li F, (2013), 'Reversible data hiding in encrypted images by reserving room before Encryption', IEEE Transactions on Information Forensics and Security, Vol.8, No.3, pp.553-562.

[10] Shim H.J, Ahn J, Jeon B, (2004), 'DH-LZW Data hiding in the Compression codes', In Proceedings of the International conference on image processing, Vol.2, No.4, pp.2195–2198.

[11] Tai W.L, Yeh C.M, Chang C.C, (2009), 'Reversible Data Hiding based on Histogram modification of pixel differences', IEEE Transactions on Circuits and Systems for Video technology, Vol.19, No.6, pp.906-910.

[12] Tseng H.W, Chang C.C, (2008), 'An Extended Difference expansion algorithm foe reversible watermarking', Image and Vision Computing, Vol.26, No.8, pp.1148-1153.

[13] Wang Z.H, Yang H.R, Cheng T.F, (2013), 'High Performance reversible data hiding scheme for LZW codes', Journal of Systems and Software, Vol.86, No.3, pp 2771-2778.