An Improved Reversible Data-Hiding Scheme for LZW Codes

Wenqiang Zhao [a], Bailong Yang [b], Shizhong Gong [c], Xuesong Li [d]

Xi'an high-tech Institution, Xi'an 710025, China

[a]qqingnine@163.com, [b]yangbl_68@163.com, [c]gszmeself@163.com, [d]lixsnzz@163.com

Abstract. HPDH-LZW is a reversible data hiding scheme based on Lempel-Ziv-Welch (LZW) compression codes. It could achieve better hiding capacity by increasing the number of symbols available to hide secrets and faster hiding and extracting speeds due to the lower computation requirements. HPDH-LZW scheme embeds secret data in LZW compression codes by modifying the value of the compression codes, but the modified LZW compression codes is not continuous so that its data space could not be fully used. To solve this problem, we subdivide the compression codes data space and hide different number of secret data in different subspaces. Guided by the idea we realize an improved reversible data-hiding scheme for LZW codes. Experiments show that the scheme we proposed has greater data hiding capacity without increasing the size of the compression file and computational complexity when compared with HPDH-LZW scheme.

Keywords: Data Hiding; LZW Compression; Compression Codes.

1. Introduction

With the extensive application of multimedia technology and rapid development of Internet, the phenomenon that huge amounts of data are transmitted over the Internet becomes widespread. It is necessary to encrypt sensitive data before transmission to protect those data in the face of a growing number of security threats. Reversible data-hiding techniques [1, 2] provide a solution which can ensure that the receiver can receive hidden messages and recover carrier data without distortion.

In a reversible data hiding scheme, the secret data are embedded in a reversible way so that one can not only extract the hidden data but also restore the original cover media without any distortion. Because of its good invisibility and high data hiding capacity, it can effectively protect the sensitive data, which makes it be widely used in military, judicial, medicine and other fields. Reversible data hiding scheme is mainly divided into spatial domain, transformed domain and the compression domain [3-5]. The reversible data hiding scheme in the compression domain could guarantee the safety of the transmitted data and reduce the transmission costs at the same time.

LZW compression algorithm, named after Lemple, Ziv, and Welch [6], is a lossless compression method. It mainly used for image compression and has high compression rate and decompression speed. With the help of a established string table, LZW algorithm could denote a long string by a short code, so it is called "string table compression algorithm as well". In the compression process, LZW algorithm gradually and dynamically generates corresponding relationships of the string and compression codes, and the relationship is a "dictionary" implicited in the compressed data. In the decompression process, the "dictionary" is gradually generated to recover the data. It is not necessary to analysis the source file and send additional information to the receiver [7] in the LZW compression. Currently, various reversible data hiding schemes based on LZW have been published [8, 9].

HPDH-LZW scheme [10] (A high-performance reversible data-hiding scheme for LZW codes), is a new reversible data hiding scheme. This scheme hides data through the relationship between the value of the output compression codes and the size of the dictionary. It has much higher data hiding capacity and lower computation costs than other methods. But its compression codes are highly discrete, and its compression codes data space could not be fully used. In order to solve this problem, we propose an improved reversible data-hiding scheme for LZW codes, which has a greater data hiding capacity without increasing the size of the compression file and computational complexity when compared with HPDH-LZW scheme.

The capacity, one of the most important properties of data hiding schemes, is increased in the proposed scheme. The rest of this paper is organized as follows: In Section 2, the LZW compression

and HPDH-LZW scheme are briefly described repectively. The proposed scheme is introduced in Section 3, and Section 4 illustrates the experimental results. In Section 5 we summarize the contributions of our scheme.

2. Related works

2.1 The LZW algorithm

In LZW algorithm, the root dictionary is initialized with ASCII values from 0 to 255 and expansion dictionary is created when the program is running. Whenever a character is input, LZW will check whether the new string formed by the old character (string) and newly input character is included in the dictionary; if so, it continue to read the next character; if not, it will add the new string to the dictionary, output the index of the old character (string) in dictionary, and assign the newly input character to the old character (string). The above process will be performed until entire file is read. When decoding, the dictionary is dynamically and automatically generated as well, LZW codes are converted to the corresponding string according to the dictionary.

2.2 The HPDH-LZW scheme

HPDH-LZW algorithm is based on LZW algorithm. In HPDH-LZW algorithm, as long as a new dictionary entry is generated, a secret information bit could be hided. The number of hidden secret bits is the same as the number of newly generated dictionary entries.

(1)The hiding algorithm of HPDH-LZW:

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.
If b='1'
Set C=C+Size, where Size is the current size of dictionary.

Step 3 Output C and add s into dictionary.
Set $s = c_s$, where c_s is the last character of s.
Continue to Step 2.

(2)The data-extracting algorithm of HPDH-LZW:

Input: LZW codes.

Output: Source file and secret file.

Step 1 Get a LZW code C_0'.
If $C_0' >$ Size′, where Size′ is the current dictionary size,
Set $C_0'=C_0'$-Size′.
Secret bit='1'.
Else
Secret bit='0'.
Output s′, where s′ is the symbol of C_0' in the dictionary.
Get character c_s', where c_s' is the first character of s′.
Set $O'= C_0'$, where the O′ is the old code.

Step 2 Get the next LZW code C′.
If C′>Size′+1
Set C′=C′-Size′-1.
Secret bit='1'.

Else,
Secret bit='0'.
If C′ is not in the dictionary,
Set s′=s_o′||c_s′, where s′ is a string variable, and s_o′ is the symbol of O′ in the dictionary.
Else,
Set s′= s_o′, where s_o′ is the symbol of O′ in the dictionary.
Output s′.
Set s_{new}′= s_o′||c_s′, where s_{new}′ is a new symbol, s_o′ is the symbol of O′ in the dictionary, and c_s′ is the first character of s′.
Add s_{new}′ into the dictionary.
Step 3 Set O′=C′ and continue to Step 2.

3. The proposed scheme

The main idea of the HPDH-LZW scheme is to modify the value of LZW code to hide data, then we could hide a secret bit when a new dictionary is generated. If the secret bit is '1', the output LZW code is the sum of the value of the LZW code and the size of the current dictionary. In this case, the modified output LZW codes are high discrete and the compression codes data space may not be fully used. In this paper, we subdivide the compression codes data space and hide different number of secret data in different subspaces and the problem is improved. When the size of dictionary is less than or equal to 384, the information hiding and extraction process are introduced in Section 3.1 and 3.2 respectively; when the dictionary size is greater than 384, hiding and extraction process is the same as HPDH-LZW algorithm.

3.1 The hiding algorithm of proposed scheme:

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.
If b='1'
If C>255
Set C= Size + (Size –C), where Size is the current size of dictionary.
Else Get next secret bit b_{next}
If 'b_{next}'=1
Set C=C+768;
Else
Set C=C+512, return b_{next} to the source file.
Else C remains the same.
Step 3 Output C and add s into dictionary.
Set s = c_s, where c_s is the last character of s.
Continue to Step 2.

3.2 The data-extracting algorithm of proposed scheme:

Input: LZW codes.
Output: Source file and secret file.
Step 1 Get a LZW code C_0′.

Let ItemPt be the current dictionary size.

If $C_0' \in$ [ItemPt, 512), get Secret bit='1', Set $C_0' = (ItemPt+1)*2-C_0'$;

If $C_0' \in$ [512, 768), get Secret bit='1', Set $C_0' = C_0'-512$;

If $C_0' \in$ [768, 1024), get Secret bit='11', Set $C_0' = C_0'-768$;

Else get a Secret bit='0'.

Output s′, where s′ is the symbol of C_0' in the dictionary.

Get character c_s', where c_s' is the first character of s′.

Set $O' = C_0'$, where the O′ is the old code.

Step 2 Get the next LZW code C′.

If $C' \in$ [ItemPt, 512), get Secret bit='1', Set $C' = (ItemPt+1)*2-C'$;

If $C' \in$ [512, 768), get Secret bit='1', Set $C' = C'-1-512$;

If $C' \in$ [768, 1024), get Secret bit='11', Set $C' = C'-1-768$;

Else get a Secret bit='0'.

If C′ is not in the dictionary,

Set $s' = s_o' || c_s'$, where s′ is a string variable, and s_o' is the symbol of O′ in the dictionary.

Else

Set $s' = s_o'$.

Output s′.

Set $s_{new}' = s_o' || c_s'$, where s_{new}' is a new symbol, s_o' is the symbol of O′ in the dictionary, and c_s' is the first character of s′.

Add s_{new}' into the dictionary.

Step 3 Set $O' = C'$ and continue to Step 2.

4. Experimental results

We use the texts and the pictures in the HPDH-LZW scheme as the test object. Text test objects are: Lincoln's Gettysburg address ("Lincoln.txt"), Obama's speech ("Obama.txt"), a smaller size research paper ("paper-s.txt"), and a larger size research paper ("paper-b.txt"). 10 256pix x 256pix grayscale test images are Baboon, Boat, Goldhill, Jet (F16), Lena, Pepper, SailBoat, Tiffany, Toys, and Zelda. The secret data file used in the experiment is generated by random generator. The length of each LZW compression code is 10 bits, and the size of the initial dictionary and the additional dictionary are both 256.

As shown in Table 1 and Table 2, data hiding capacity of our improved algorithm for texts and images except Paper-s.txt and Jet have increased. In average, the hiding capacity of our algorithm for text was increased by 7.77% and that for image was increased by 12.34%. The hiding capacity of Paper-s.txt is 25574 in the HPDH-LZW scheme, and we consider this data is incorrect.

Table 1 Comparison to HPDH-LZW scheme for text data

File name	Original size	HPDH-LZW Hidden bit(A)	Proposed scheme Hidden bit(B)	Increased bits (B-A)/A
Obama.txt	13504	8856	9529	7.60%
Lincoln.txt	1450	912	998	9.43%
Paper-b.txt	80316	50939	54138	6.28%
Paper-s.txt	20079	25574	13678	

Table 2 Comparison to HPDH-LZW scheme for image data

File name	Original size	HPDH-LZW Hidden bit(A)	proposed scheme Hidden bit (B)	Increased bits (B-A)/A
Baboon	65536	63844	70861	10.99%
Boat	65536	52980	61416	15.92%
Goldhill	65536	59577	67450	13.21%
Jet	65536	52090	49056	-5.82%
Lena	65536	61437	67153	9.30%
Pepper	65536	60839	68637	12.82%
SailBoat	65536	58498	67367	15.16%
Tiffany	65536	56627	66913	18.16%
Toys	65536	49572	52396	5.70%
Zelda	65536	60771	66722	9.79%

As we can see in Table 1 and Table 2, the data hiding capacity for images is lager than that of text. This is because the text data is constituted by ASCII code 0-127 and image pixel value range in 0-255 so that the improved scheme may output more root dictionary representation of the character on image data. Hiding more continuous secret bit 1 in the output of root dictionary output is only suitable for the situation that the size of extended dictionary is small (for example, in this paper, the size of the root dictionary and the extended dictionary is 256). If the size of the extended dictionary is larger than root dictionary, we may infer that more continuous secret bit 1 are hidden in the output of extended dictionary. Our method provides a way to increase the amount of data hiding, but specific solutions are needed in the face of specific issues.

The improved scheme optimizes the LZW encoding method, therefore it can hide more secret data. At the same time, they have same method in the dictionary generation, the size of the compression file and computational complexity is not increased. The experimental results show that the improved scheme is effective.

5. Conclusion

In this paper, an improved reversible data hiding scheme based on LZW compression coding is proposed. Through the rational division and the full use of the data space that the 10 bit codes could represent, the improved scheme can hide more secret data. The improved scheme and the original scheme share the same method in the dictionary generation, so our scheme does not increase the size of the compression file and computational complexity. The experimental show that the scheme we proposed has greater data hiding capacity.

References

[1].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 19(6), 906–910.

[2].Tseng, H.W., Chang, C.C., 2008. An extended difference expansion algorithm for reversible watermarking. Image and Vision Computing 26(8), 1148–1153.

[3].Chang, C.C., Wu, W.C., 2005. A steganographic method for hiding secret data using side match vector quantization. IEICE Transactions on Information and Systems E88–D(9), 2159–2167.

[4].Chen, W.J., Huang, W.T., 2009. VQ indexes compression and information hiding using hybrid lossless index coding. Digital Signal Processing 19(3), 433–443.

[5].Jo, M., Kim, H.D., 2002. A digital image watermarking scheme based on vector quantisation. IEICE Transactions on Information and Systems E85-D(6), 1054–1056.

[6].Gonzalez, R.C., Woods, R.E., 2002. Digital Image Processing, 2nd Edition, Prentice Hall, Jan.

[7].Nandi, U., Mandal, J.K., 2012. A compression technique based on optimality of LZW code (OLZW). In: 2012 Third International Conference on Computer and Communication Technology (ICCCT), Allahabad, pp.166–170.

[8].Shim, H.J., Ahn, J., Jeon, B., 2004. DH-LZW: lossless data hiding in LZW compression. In: Proceedings of the International Conference on Image Processing, 4, Singapore, pp.2195–2198.

[9].Chen, C.C., Chang, C.C., 2010. High-capacity reversible data-hiding for LZW codes. In: Proceedings of the 2nd International Conference on Computer Modeling and Simulation, 1, Sanya, China, pp.3–8.

[10].Wang, Z.H., Yang, H.R., Cheng, T.F., Chang, C.C., 2013. A high-performance reversible data-hiding scheme for LZW codes. The Journal of Systems and Software 86(11), pp.2771-2778.