Cao *et al. SpringerPlus (2016) 5:175*
DOI 10.1186/s40064-016-1818-0

SpringerPlus
a SpringerOpen Journal

RESEARCH Open Access

High capacity data hiding scheme based on (7, 4) Hamming code

Zekun Cao[1], Zhaoxia Yin[1,2*], Honghe Hu[1], Xiangping Gao[1] and Liangmin Wang[1]

*Correspondence:
yinzhaoxia@ahu.edu.cn
[1] Key Laboratory of Intelligent Computing & Signal Processing, School of Computer Science & Technology, Anhui University, Hefei 230601, Anhui, China
Full list of author information is available at the end of the article

Abstract

Aiming to embed large amount of data while minimize the sum of costs of all changed pixels, a novel high capacity data hiding scheme based on (7, 4) Hamming code is realized by a family of algorithms. Firstly, n ($n = 1, 2, 3$) cover pixels are assigned to one set according to the payload. Then, 128 binary strings of length seven are divided into eight sets according to the syndrome of every binary string. Binary strings that share the same syndrome are classified into one set. Finally, a binary string in a certain set determined by the data to be embedded is chosen to modify some of the least significant bits of the n cover pixels. The experimental results demonstrate that the image quality of the proposed method with high embedding payload is superior to those of the related schemes.

Keywords: Data hiding, Hamming code, Image quality, Embedding capacity

Background

Data hiding, frequently interchangeably referred to as information hiding, is the art of embedding additional data in a certain carrier (Zielińska et al. 2014). These carriers are typically digital media files transmitted on the Internet, such as images, audios, videos, or text (Ker et al. 2013). Historically, the design of data hiding schemes for digital images has heavily relied on heuristic principles (Feng et al. 2015; Hong et al. 2015; Qian and Zhang 2015; Xia et al. 2014a, b). The current trend calls for constraining the embedding changes to image segments with complex content. Such adaptive data hiding schemes are typically realized by first defining the cost of changing each pixel and then embedding the additional data while minimizing the sum of costs of all changed pixels. One of the explorations to achieve this goal is applying the error correcting code to data hiding, and many researchers have done a lot of research in this area (Chang and Chou 2008; Chen et al. 2013; Liu 2007; Ma et al. 2013; Wang 2009; Yin et al. 2010; Zhang et al. 2007; Zhu et al. 2010).

Crandall originally proposed a data hiding scheme named matrix encoding (Crandall 1998) in 1998. In this scheme, k bits were embedded into $2^k - 1$ cover pixels by modifying the least significant bits (LSBs) of one pixel. The embedding capacity reached $k/(2^k - 1)$ bit per pixel (bpp). Based on the matrix encoding, Zhang et al. (2007) proposed the "Hamming+1" scheme in 2007. Compared with the matrix encoding scheme, it used one more cover pixel to embed one more bit while the cost remain unchanged. Thus, the

embedding capacity got increased to be $(k + 1)/2^k$ bpp. Later, Chang et al. proposed a new scheme (Chang and Chou 2008) based on the idea of classification in 2008. Binary strings were assigned into eight sets. A binary string of length $2^k - 1$ in a specific set was selected out to embed k bits. It presented a new idea in applying Hamming code to data hiding. But the embedding capacity didn't get improved compared with the previous two scheme. It is equal to the embedding capacity of the matrix encoding scheme (Crandall 1998).

The marked-image quality of the aforementioned schemes is ideal when the embedding payload is low (no more than $k/(2^k - 1)$ or $(k + 1)/2^k$ bpp), but it degrades hardly with the increase of the embedding payload. Against this problem, a new data hiding scheme based on (7, 4) Hamming code is proposed in this paper. The marked-image quality of the proposed scheme is superior to those of the related works in Crandall (1998), Zhang et al. (2007) and Chang and Chou (2008) under a high embedding payload.

Related works

The Hamming code

An error correcting code could not only detect that errors have occurred but also locate the error positions. Hamming code is a linear error correcting code that can detect and correct single bit errors. The $(n, n - k)$ Hamming code uses n cover bits to transmit $n - k$ message bits, and the other k bits used for error correcting purpose are called parity check bits, where $n = 2^k - 1$ on the binary filed.

$S = \{C_1, C_2, ...,C_M\}$ is a set of code words. The number of elements of S, denoted as $|S|$, is called the cardinality of the code. For any two code words $x = (x_1, x_2, ..., x_n) \in S$ and $y = (y_1, y_2, ..., y_n) \in S$, the Hamming distance is defined by $d_H(x, y) = |\{i|x_i \neq y_i\}|$. The minimum distance of the code S is defined as $d_{\min} = \min\{d_H(x, y)|x, y \in S\}$. And the covering radius of the code S is r if any binary string $u = (u_1, u_2, ..., u_n)$ differs from at least one code word $x = (x_1, x_2, ..., x_n) \in S$ in at most r positions. The minimum distance $d_{\min}$ measures the error-correcting capability, and the maximum distortion that occurs when a binary string is replaced by a proper code word is measured by the covering radius r. Therefore, a large value of the minimum distance $d_{\min}$ is preferable to the purpose of error correction whereas a small value of the covering radius r is preferable to the purpose of steganography. The (7, 4) Hamming code is a binary code of length $n = 7$, with cardinality $|S| = 16$, minimum distance $d_{\min} = 3$, and covering radius $r = 1$.

The (7, 4) Hamming code is now taken as an example to demonstrate how Hamming code correct an error bit. Suppose that the message bits are $m = (1010)$. First, the code generator matrix $\boldsymbol{G}$ is used to form n cover bits C as follows.

$$C = m \times \boldsymbol{G} = (1010) \times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = (1010010)$$

Next, the code word C is transmitted to a receiver via a noise communication channel. Supposed that the received code word is $C' = (1011010)$. Then the parity check matrix $\boldsymbol{H}$ is used to compute the syndrome vector $\boldsymbol{z} = (z_1, z_2, z_3)$ for checking an error as follows.

$$\boldsymbol{z} = (H \times C'^{\mathrm{T}})^{\mathrm{T}} = \left(\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \times (1011010)^{\mathrm{T}} \right)^{\mathrm{T}} = (011)$$

The vector $\boldsymbol{z}^{\mathrm{T}} = (011)^{\mathrm{T}}$ is identical to the fourth column of the parity check matrix $\boldsymbol{H}$. Thus, an error is detected at the fourth position of C', and C' is corrected by $C' = C' \oplus e_4 = (1010010)$, where $\oplus$ is the exclusive-or operation, and e_i, the error pattern, is a unit vector of length n with a "1" located at the i-th position. If the syndrome vector is $\boldsymbol{z} = (000)$, the receiver can conclude that no error has occurred.

"Matrix Encoding"

In the matrix encoding scheme, a string of k bits $\mathbf{s} = (s_1, s_2, \ldots, s_k)$ is embedded into a group of n cover pixels by adding or subtracting one to or from at most one cover pixel, where $n = 2^k - 1$. Firstly, the syndrome vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_k)$ is calculated by $\boldsymbol{z} = (\mathbf{c} \times \boldsymbol{H}^{\mathrm{T}}) \oplus \mathbf{s}$, with $\mathbf{c} = (\mathrm{LSB}(p_1), \mathrm{LSB}(p_2), \ldots, \mathrm{LSB}(p_n))$ and LSB (p_i) means the least significant bit of i-th pixel p_i. $\boldsymbol{H}$ is the parity check matrix of the $(n, n-k)$ Hamming code. T is the transpose operation, and $\oplus$ is the exclusive-or operation. Next, if the computed syndrome vector $\boldsymbol{z}$ is (0, 0,..., 0), then the group of n marked pixels $\boldsymbol{R}$ is set to be equal to $\mathbf{c}$; otherwise, find the i-th column of $\boldsymbol{H}$ that is equal to the transposed syndrome vector $\boldsymbol{z}^{\mathrm{T}}$. The group of n marked pixels $\boldsymbol{R}$ is calculated by $\boldsymbol{R} = \mathbf{e}_i \oplus \mathbf{c}$, where $\mathbf{e}_i$ is a unit vector of length n with "1" located at the i-th position. At the receiving side, a receiver can extract the original binary string $\mathbf{s}$ from the received group $\boldsymbol{R}$ by $\mathbf{s} = \boldsymbol{R} \times \boldsymbol{H}^{\mathrm{T}}$.

"Hamming+1"

Zhang et al. proposed the "Hamming+1" scheme (Zhang et al. 2007) to embed a string of $(k+1)$ secret bits $\mathbf{s} = (s_1, s_2, \ldots, s_{k+1})$ into a group of $(n+1)\psi$ cover pixels $\boldsymbol{p} = (p_1, p_2, \ldots, p_{n+1})$, where $n = 2^k - 1$, by modifying at most one cover pixel as follows.

$$(s_1, s_2, \ldots, s_k) = (\mathrm{LSB}(p_1), \mathrm{LSB}(p_2), \ldots \mathrm{LSB}(p_n)) \times \boldsymbol{H}^{\mathrm{T}}) \tag{1}$$

$$s_{k+1} = \left(\left\lfloor \frac{p_1}{2} \right\rfloor + \left\lfloor \frac{p_2}{2} \right\rfloor + \cdots + \left\lfloor \frac{p_n}{2} \right\rfloor + p_{n+1} \right) \bmod 2 \tag{2}$$

where $\boldsymbol{H}$ is the parity check matrix of the $(n, n-k)$ Hamming code, T is the transpose operation. This means that the first $k\psi$ secret bits of $\mathbf{s}$ are embedded into the first n bits of $\boldsymbol{p}$ by using matrix encoding, and the last secret bit of $\mathbf{s}$ is embedded by using the function of $n\psi$ cover pixels $\boldsymbol{p}$. The embedding rules proposed in (Zhang et al. 2007) are as follows. If Eq. (1) does not hold, then $\boldsymbol{p}_{n+1}$ is kept unchanged, and one cover pixel $\boldsymbol{p}_i$ $(1 \le i \le n)$ needs to be increased or decreased by one to make Eqs. (1) and (2) hold simultaneously. If (1) holds and (2) does not, the first n pixels are kept unchanged and last cover pixel $\boldsymbol{p}_{n+1}$ is randomly increased or decreased by one.

At the receiving side, a receiver can extract the first $k\psi$ secret bits of $\mathbf{s}$ by applying the extracting way of the matrix encoding scheme and the last secret bit of $\mathbf{s}$ can be extracted by using Eq. (2).

"Nearest Code"

In the nearest covering code scheme (Chang and Chou 2008), all possible combinations of seven bits are classified into eight sets G_0, G_1, ...G_7. There are 16 elements $\boldsymbol{G}_u^0, \boldsymbol{G}_u^1, \ldots \boldsymbol{G}_u^{15}$ in each set G_u, where $0 \leq u \leq 7$. And $\boldsymbol{G}_u^v$ satisfies equation $u = \boldsymbol{G}_u^v \times \boldsymbol{H}^{\mathrm{T}}$, where $0 \leq v \leq 15$, $\boldsymbol{H}$ is the parity check matrix of the (7, 4) Hamming code, T is the transpose operation. A covering code $\boldsymbol{G}_s^v$ with nearest Hamming distance to $\boldsymbol{P} = (\mathrm{LSB}(p_1), \mathrm{LSB}(p_2), \ldots, \mathrm{LSB}(p_7))$ is selected in $\boldsymbol{G}_s$ according to secret bits $\mathbf{s} = (s_1, s_2, s_3)$, where the subscript of $\boldsymbol{G}_s$ is equal to the corresponding decimal number of $\mathbf{s} = (s_1, s_2, s_3)$. Then, the cover pixels are modified by $\boldsymbol{G}_s^v$. At the receiving side, a legal receiver can extract the original secret bits $\mathbf{s}$ from the received group of 7 pixels $\boldsymbol{R}$ by $\mathbf{s} = \boldsymbol{R} \times \boldsymbol{H}^{\mathrm{T}}$.

The proposed scheme

In the proposed scheme, a secret binary string of length three is in a mapping relationship with the error pattern of the (7, 4) Hamming code and then can be embedded into a group of cover pixels. The number of the cover pixels in different groups varies under different embedding payload.

The preparations

I is the cover image sized $H \times W$, and *marked_I* is the marked-image with data $D = \{d_1, \ldots, d_L\}$embedded, where $d_i \in \{0, 1\}$, $1 \leq i \leq L$. $\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$ is a parity check matrix of the (7, 4) Hamming code. A string of binary bits $(b_1 b_2 \ldots b_7)$ is the cover of a string of three bits $(d_i d_{i+1} d_{i+2})$, and $(b_1' b_2' \ldots b_7')$ is the marked-string of $(b_1 b_2 \ldots b_7)$. p_i is the i-th pixel in cover image, and p_i' is the i-th pixel in marked-image. p_i^j represents the j-th least significant bit of pixel p_i. ER, i.e. embedding rate, is calculated as follows.

$$\mathrm{ER} = \frac{L}{H \times W} \mathrm{bpp} \tag{3}$$

N_n is the number of groups that n ($n = 1, 2, 3$) cover pixels are used to embed a three bits string. And N_n satisfies Formula (4). The first equation of Formula (4) indicates that the number of bits to be embedded is equal to the amount of bits the cover image could bear under a particular embedding rate. And the second equation in Formula (4) requires that the cover pixels we need are less or equal to the pixels the cover image could provide.

$$\begin{cases} 3(N_1 + N_2 + N_3) = H \times W \times ER \\ N_1 + 2N_2 + 3N_3 \leq H \times W \end{cases} \tag{4}$$

To modify the cover pixels as less as possible, Formula (4) is processed to obtain Formula (5) based on the following considerations. The top priority scheme, grouping three cover pixels together to embed a three bits string, satisfies Formula (4) when $0 \leq ER \leq 1$. When $1 < ER \leq 1.5$, grouping two cover pixels to embed a binary string satisfies Formula (4), but there will be some pixels in the cover image unused. Instead, we embed some secret binary strings into groups of three cover pixels and the others into groups of two cover pixels. Obviously, this scheme causes less modification to cover image than the

scheme that only using two cover pixels to embed binary strings. Likewise, we embed some binary strings into groups of two cover pixels and the others into groups of one cover pixel when $1.5 < ER < 3$. Therefore, adaptive N_n is calculated by Formula (5), which contributes to minimize the sum of costs of all changed pixels.

$$\begin{cases} N_1 = 0, N_2 = 0, N_3 = \frac{L}{3} & (0 < \mathrm{ER} \le 1) \\ N_1 = 0, N_2 = L - H \times W, N_3 = H \times W - \frac{2L}{3} & (1 < \mathrm{ER} \le 1.5) \\ N_1 = \frac{2L}{3} - H \times W, N_2 = H \times W - \frac{L}{3}, N_3 = 0 & (1.5 < \mathrm{ER} \le 3) \end{cases} \tag{5}$$

The data embedding phase

All binary strings of length seven are classified into eight sets G_0, G_1, ...G_7. There are 16 elements in every set $\boldsymbol{G}_u = \{\boldsymbol{G}_u^v\}_{v=0}^{15}$, and $\boldsymbol{G}_u^v = (c_1, c_2, \ldots, c_7)$ satisfies equation $u = \boldsymbol{G}_u^v \times \boldsymbol{H}^{\mathrm{T}}$, where $0 \le u \le 7$. Specific embedding algorithms are as follows.

Algorithm 1: Internal embedding algorithm

Input: $(b_1 b_2 \ldots b_7)$, a string of 3 data bits $(d_i d_{i+1} d_{i+2})$

Output: $\left(b_1' b_2' \ldots b_7'\right)$

Step 1: Find a $\boldsymbol{G}_u^v$ which satisfies $(c_1 c_2 c_3 c_4) = (b_1 b_2 b_3 b_4)$ in $\boldsymbol{G}_u$, where $u = (d_i d_{i+1} d_{i+2})$;

Step 2: $\left(b_1' b_2' \ldots b_7'\right) = \boldsymbol{G}_u^v$.

Algorithm 2: External embedding algorithm

Input: *I, D*

Output: *marked_I*

Step 1: Calculate *ER* according to Eq. (3) and N_1, N_2, N_3 by Formula (5);

Step 2: $(b_1 b_2 \ldots b_7) = (p_i^7 p_i^6 \ldots p_i^1)$, then call Algorithm 1 to get $(b_1' b_2' \ldots b_7')$, and $p_i' = (p_i^8 b_1' b_2' \ldots b_7')$, $i \in \{1, 2, 3, \ldots, N_1\}$;

Step 3: $(b_1 b_2 \ldots b_7) = (p_i^4 p_i^3 p_{i+1}^3 p_i^2 p_{i+1}^2 p_i^1 p_{i+1}^1)$, then call Algorithm 1 to get $(b_1' b_2' \ldots b_7')$, and $p_i' = (p_i^8 p_i^7 p_i^6 p_i^5 b_1' b_2' b_4' b_6')$, $p_{i+1}' = (p_{i+1}^8 p_{i+1}^7 p_{i+1}^6 p_{i+1}^5 p_{i+1}^4 b_3' b_5' b_7')$, $i \in \{N_1 + 1, N_1 + 3, N_1 + 5, \ldots, N_1 + 2N_2 - 1\}$;

Step 4: $(b_1 b_2 \ldots b_7) = (p_i^3 p_i^2 p_{i+1}^2 p_{i+2}^2 p_i^1 p_{i+1}^1 p_{i+2}^1)$, then call Algorithm 1 to get $(b_1' b_2' \ldots b_7')$, and $p_i' = (p_i^8 p_i^7 p_i^6 p_i^5 p_i^4 b_1' b_2' b_5')$, $p_{i+1}' = (p_{i+1}^8 p_{i+1}^7 p_{i+1}^6 p_{i+1}^5 p_{i+1}^4 p_{i+1}^3 b_3' b_6')$, $p_{i+2}' = (p_{i+2}^8 p_{i+2}^7 p_{i+2}^6 p_{i+2}^5 p_{i+2}^4 p_{i+2}^3 b_4' b_7')$, $i \in \{N_1 + 2N_2 + 1, N_1 + 2N_2 + 4, \ldots, N_1 + 2N_2 + 3N_3 - 2\}$.

Example: data embedding

An example is now given to demonstrate the embedding phase of the proposed scheme. Suppose that I is a grayscale image with $H \times W = 3 \times 3$ shown in Fig. 1 and $\mathrm{D} = \{d_1, d_2, d_3, \ldots, d_{10}, d_{11}, d_{12}, \ldots, d_{18}\} = \{1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1\}$.

Step 1: We could calculate ER = 2, then work out $N_1 = 3$, $N_2 = 3$, $N_3 = 0$ by Formula (5).

Step 2: From the cover image, we know $p_1 = 162$, $(b_1 b_2 \ldots b_7) = (p_1^7 p_1^6 \ldots p_1^1) = (0100010)$, $(d_1 d_2 d_3) = (101)$. Then call Algorithm 1 to get $(b_1' b_2' \ldots b_7') = \boldsymbol{G}_5^v = (0100011)$, so $p_1' = (p_i^8 b_1' b_2' \ldots b_7') = 163$. Repeat Step 2 $(N_1 - 1)$ times to embed $(d_4 d_5 d_6)$ into $p_2 = 164$ and $(d_7 d_8 d_9)$ into $p_3 = 165$.

Step 3: $(p_4, p_5) = (162, 153)$, $(b_1 b_2 \ldots b_7) = \left(p_4^4 p_4^3 p_5^3 p_4^2 p_5^2 p_4^1 p_5^1\right) = (0001101)$, $(d_{10} d_{11} d_{12}) = (010)$. Call Algorithm 2 to get $(b_1' b_2' \ldots b_7') = \boldsymbol{G}_2^v = (0001001)$,

162	164	155
162	153	155
164	150	152

Fig. 1 Cover image *I*

so $p'_4 = \left(p_4^8 p_4^7 p_4^6 p_4^5 b'_1 b'_2 b'_4 b'_6\right) = 162$, $p'_5 = (p_5^8 p_5^7 p_5^6 p_5^5 p_5^4 b'_3 b'_5 b'_7) = 151$. Repeat Step 3 $(N_2 - 1)$ times to embed $(d_{13}d_{14}d_{15})$ into $(p_6, p_7) = (155, 164)$ and $(d_{16}d_{17}d_{18})$ into $(p_8, p_9) = (150, 152)$. Finally, we get the marked-image *marked_I* shown in Fig. 2.

The data extracting phase

Algorithm 3: Data extracting algorithm:

Input: *marked_I*, ER

Output: *D*

Step 1: Calculate the value of $N_n (n = 1, 2, 3)$ by Formula (5) according to ER.

Step 2: $(b'_1 b'_2 \ldots b'_7) = \left(p_i^{4'} p_i^{3'} p_{i+1}^{3'} p_i^{2'} p_{i+1}^{2'} p_i^{1'} p_{i+1}^{1'}\right)$, $(d_i d_{i+1} d_{i+2}) = (b'_1 b'_2 \ldots b'_7) \times \boldsymbol{H}^{\mathrm{T}}$, $i \in \{1, 2, \ldots, N_1\}$.

Step 3: $(b'_1 b'_2 \ldots b'_7) = \left(p_i^{4'} p_i^{3'} p_{i+1}^{3'} p_i^{2'} p_{i+1}^{2'} p_i^{1'} p_{i+1}^{1'}\right)$, $(d_i d_{i+1} d_{i+2}) = (b'_1 b'_2 \ldots b'_7) \times \boldsymbol{H}^{\mathrm{T}}$, $i \in \{N_1 + 1, N_1 + 3, N_1 + 5, \ldots, N_1 + 2N_2 - 1\}$.

Step 4: $(b'_1 b'_2 \ldots b'_7) = \left(p_i^{3'} p_i^{2'} p_{i+1}^{2'} p_{i+2}^{2'} p_i^{1'} p_{i+1}^{1'} p_{i+2}^{1'}\right)$, $(d_i d_{i+1} d_{i+2}) = (b'_1 b'_2 \ldots b'_7) \times \boldsymbol{H}^{\mathrm{T}}$, $i \in \{N_1 + 2N_2 + 1, N_1 + 2N_2 + 4, \ldots, N_1 + 2N_2 + 3N_3 - 2\}$.

Example: data extracting

Suppose the receiver receives the marked-image sized $H \times W = 3 \times 3$ shown in Fig. 2 and knows that the embedding rate (ER) is 2 bpp.

Step 1: Work out $N_1 = 3, N_2 = 3, N_3 = 0$ by Formula (5).

Step 2: $p'_1 = 163$, $(b'_1 b'_2 \ldots b'_7) = (p_1^{7'} p_1^{6'} \ldots p_1^{1'}) = (0100010)$, $(d_1 d_2 d_3) = (b'_1 b'_2 \ldots b'_7) \times \boldsymbol{H}^{\mathrm{T}} = (101)$. Repeat Step 2 $(N_1 - 1)$ times to extract the bits embedded in $p'_2 = 163, p'_3 = 158$.

163	167	158
162	151	154
165	151	152

Fig. 2 Marked-image *marked_I*

Step 3: $(p'_4, p'_5) = (162, 151)$, $(b'_1 b'_2 \ldots b'_7) = (p_3^{4\prime} p_3^{3\prime} p_4^{3\prime} p_3^{2\prime} p_4^{2\prime} p_3^{1\prime} p_4^{1\prime}) = (0001001)$, $(d_{10} d_{11} d_{12}) = (b'_1 b'_2 \ldots b'_7) \times \boldsymbol{H}^{\mathrm{T}} = (010)$. Repeat Step 3 $(N_2 - 1)$ times to extract the bits embedded in $(p'_6, p'_7) = (154, 165)$, $(p'_8, p'_9) = (151, 152)$.

Experiment results

To evaluate the performance of the proposed scheme, we simulate the "Matrix Encoding" (Crandall 1998), the "Hamming+1" (Zhang et al. 2007), the "Nearest Code" (Chang and Chou 2008) and the proposed scheme by software Matlab R2014a. Standard gray-scale test images sized 512×512 are used in the simulations, as shown in Fig. 3.

Preprocessing

In order to make comparison objectively and fairly, the embedding capacity of "Matrix Encoding" (Crandall 1998), "Hamming+1" (Zhang et al. 2007) and "Nearest Code" (Chang and Chou 2008) are also enhanced by extending the least significant bit to general LSBs. The extension method of "Matrix Encoding" is as follows. Every 3-bit string is embedded into $\boldsymbol{G}_i (1 \le i \le 7)$ which is composed of the i-th least significant bit of 7 pixels by the matrix encoding method. Thus, the embedding capacity of the extended "Matrix Encoding" method become 3 bpp.

The "Hamming+1" scheme is extended as follows. Every 4-bit string is embedded into $\boldsymbol{G}_i (1 \le i \le 4)$, composed of the $(2i - 1)$-th and $2i$-th least significant bits of 8 pixels, using the "Hamming+1" method. Thus, the embedding capacity of the extended "Hamming+1" scheme become 2 bpp.

Also, the extension method of the "Nearest Code" is as follows. Every 3-bit string is embedded into $\boldsymbol{G}_i (1 \le i \le 7)$ which is composed of the i-th least significant bit of 7 pixels by the "Nearest Code" method, making the embedding capacity of the extended "Nearest Code" method be 3 bpp.

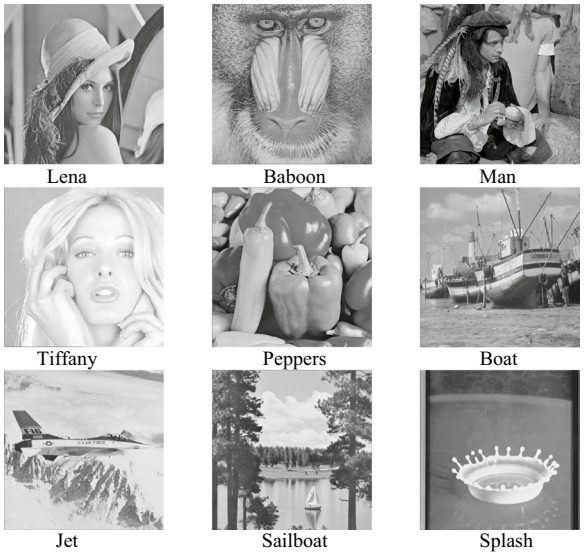

Fig. 3 The nine test images

To be fair to compare with the related works, the same method used in obtaining Formula (5) is applied here to process the extended "Matrix Encoding", "Hamming+1" and "Nearest Code" to be adaptive to the payload as follows.

The extended "Matrix Encoding":

$$\begin{cases} N_1 = L & ER \le \frac{3}{7} \\ N_i = \frac{i+1}{7}H \times W - L, N_{i+1} = L - \frac{i}{7}H \times W & \frac{3i}{7} < ER \le \frac{3(i+1)}{7} (1 \le i \le 6) \end{cases} \tag{6}$$

The extended "Hamming+1":

$$\begin{cases} N_1 = L & ER \le \frac{1}{2} \\ N_i = \frac{i+1}{8}H \times W - L, N_{i+1} = L - \frac{i}{8}H \times W & \frac{i}{2} < ER \le \frac{i+1}{2} (1 \le i \le 3) \end{cases} \tag{7}$$

The extended "Nearest Code":

$$\begin{cases} N_1 = L & ER \le \frac{3}{7} \\ N_i = \frac{i+1}{7}H \times W - L, N_{i+1} = L - \frac{i}{7}H \times W & \frac{3i}{7} < ER \le \frac{3(i+1)}{7} (1 \le i \le 6) \end{cases} \tag{8}$$

where, N_i represents the groups of data bits embedded in $\boldsymbol{G}_i$.

Image quality

PSNR (Peak Signal to Noise Ratio) is widely used to measure the image quality of marked-images by calculate the difference between the marked-image and the cover image, which is defined as follows.

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\text{MSE}}(\text{dB}) \tag{9}$$

$$\text{MSE} = \frac{1}{H \times W}\sum_{i=1}^{H}\sum_{j=1}^{W}(I_{i,j} - I'_{i,j})^2 \tag{10}$$

The above equations demonstrate that the smaller the difference between the marked-image and cover image is, the greater the PSNR value is. In general, if a marked-image with PSNR value greater than 30 dB, the distortion of the marked-image is hard to be detected by human eyes.

Tables 1, 2, 3 and 4 show the PSNR values of marked images generated by different methods with several payloads, i.e. ER = 1 bpp, ER = 1.5 bpp, ER = 2 bpp and ER = 3 bpp. The data in tables are the mean value of ten independent experiments. And

Table 1 The PSNR comparison of different methods with ER = 1 bpp

	Lena	Baboon	Man	Tiffany	Peppers	Boat	Jet	Sailboat	Splash
Matrix encoding	47.02	47.02	47.03	47.02	47.02	47.01	47.04	47.01	47.03
Nearest code	47.02	47.02	47.01	47.03	47.02	47.01	47.04	47.01	47.03
Hamming+1	45.14	45.14	45.01	45.10	45.14	45.14	45.14	45.14	45.13
Proposed scheme	51.14	51.14	51.14	51.15	51.14	51.14	51.15	51.14	51.14

Table 2 The PSNR comparison of different methods with ER = 1.5 bpp

	Lena	Baboon	Man	Tiffany	Peppers	Boat	Jet	Sailboat	Splash
Matrix encoding	39.90	39.92	39.92	39.93	39.92	39.94	39.93	39.93	39.88
Nearest code	39.90	39.91	39.92	39.93	39.91	39.94	39.93	39.93	39.88
Hamming+1	33.08	33.10	32.73	32.77	33.04	33.05	33.08	33.09	33.01
Proposed scheme	46.37	46.37	46.37	46.38	46.37	46.36	46.38	46.37	46.37

Table 3 The PSNR comparison of different methods with ER = 2 bpp

	Lena	Baboon	Man	Tiffany	Peppers	Boat	Jet	Sailboat	Splash
Matrix encoding	33.10	33.08	33.06	33.09	33.05	33.10	33.10	33.06	33.18
Nearest code	33.11	33.07	33.06	33.09	33.06	33.10	33.10	33.07	33.18
Hamming+1	20.62	20.85	20.25	19.78	20.63	20.70	19.98	20.27	20.54
Proposed scheme	41.61	41.60	41.62	41.57	41.60	41.58	41.67	41.59	41.62

Table 4 The PSNR comparison of different methods with ER = 3 bpp

	Lena	Baboon	Man	Tiffany	Peppers	Boat	Jet	Sailboat	Splash
Matrix encoding	19.80	19.77	19.63	19.87	19.89	19.70	20.07	20.09	19.82
Nearest code	19.80	19.77	19.64	19.87	19.89	19.69	20.08	20.09	19.81
Hamming+1	–	–	–	–	–	–	–	–	–
Proposed scheme	37.92	37.92	37.92	37.91	37.92	37.92	37.98	37.89	37.94

data bits embedded into images are generated randomly. From the tables, it's obvious that the PSNR values of the proposed scheme are higher than those of the related works. It indicates that the marked-image quality of the proposed scheme is superior to those of the related works under the same payload.

The PSNR-ER comparison results of Lena and Baboon are shown in Figs. 4 and 5. From the figures, the PSNR values of the proposed scheme are slightly lower than those of the extended "Matrix Encoding", "Hamming+1" and "Nearest Code" schemes when the embedding rate is relatively small, but while the embedding rate gets greater, the PSNR values of the proposed scheme are significantly higher than those of the other methods. By the way, the curves of the "Extend Matrix Encoding" and the "Extend Nearest Code" are completely overlapped, because both of the two methods embed three bits by modifying one bit. Thus, only the results of the extended "Matrix Encoding" scheme are shown in the next experiment results.

Take Lena and Baboon for examples, the marked-images of the extended "Matrix Encoding", the extended "Hamming+1"and the proposed scheme under different payloads are shown in Figs. 6 and 7. We can see that there is no distinct difference between the marked-images when the embedding rate is 3/7 bpp. When the embedding rate is up to 2 bpp, we can see spots easily on the marked-image of the extended "Hamming+1" scheme, and can hardly see any spot on the marked-image of the proposed scheme. The same observations can be found between the proposed scheme and the extended "Matrix Encoding" scheme while ER = 3 bpp.

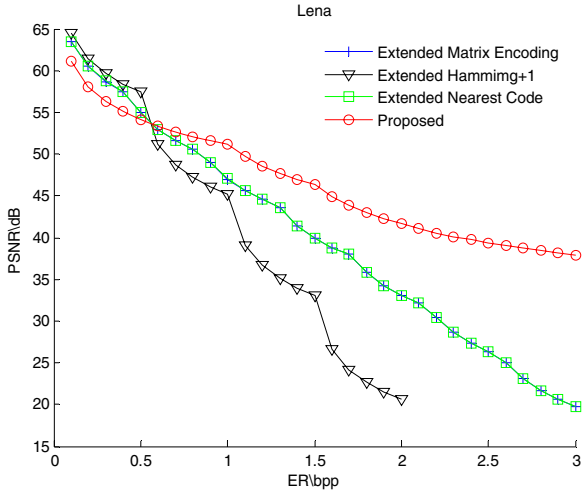

Fig. 4 PSNR-ER comparison of Lena

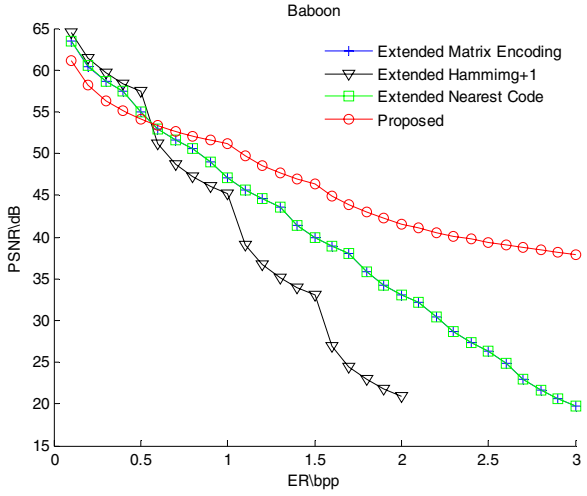

Fig. 5 PSNR-ER comparison of Baboon

Security analysis

Security is a significant problem for data hiding. Many steganalysis methods uses statistics tools to analysis the pixel value distribution on a suspicious image for cracking the secret message delivery. From this point of view, we analysis the pixel histograms between the test cover images and the marked images to measure the security of the data hiding methods. Take a smooth content image Lena and a complex content image Baboon for example, the pixel histogram results generated by the extended "Matrix Encoding" scheme, the extended "Hamming+1" scheme, and the proposed scheme with high payloads are shown in Fig. 8. From Fig. 8, the pixel histogram of the marked image generated by the proposed scheme is closer to the pixel histogram of the original image than those of the extended "Matrix Encoding" and extended "Hamming+1" scheme. It

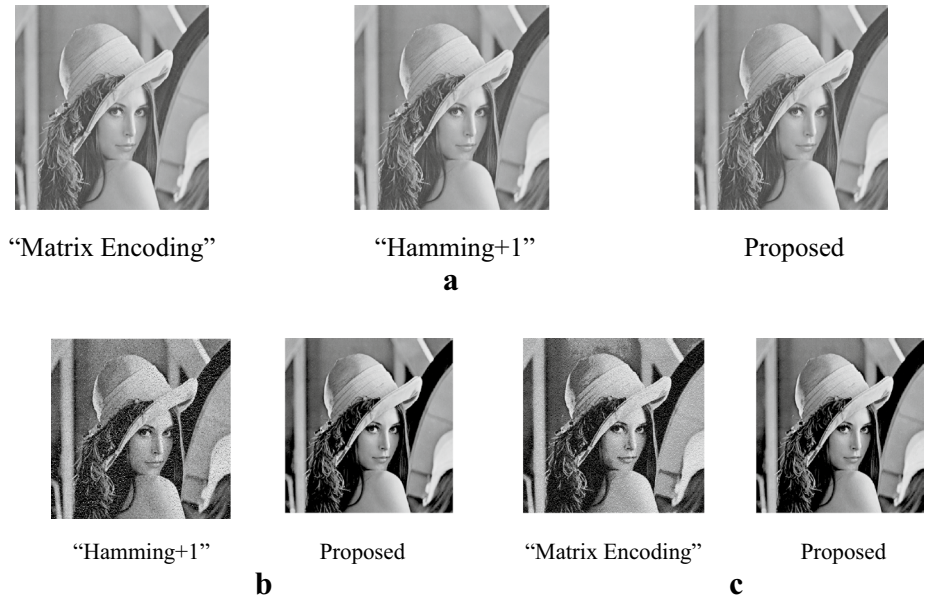

Fig. 6 Marked-images of Lena under various payloads. **a** ER = 3/7 bpp. **b** ER = 2 bpp. **c** ER = 3 bpp

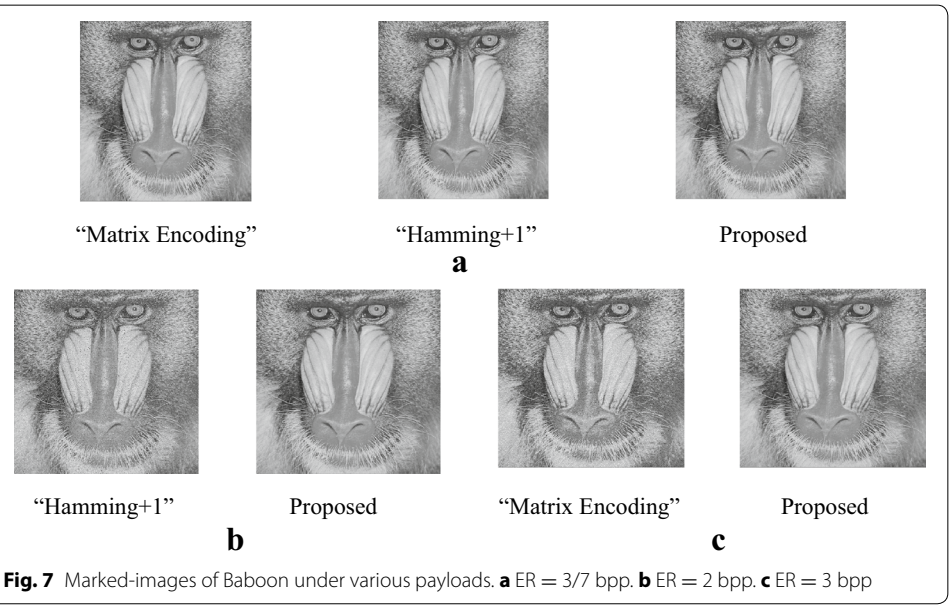

Fig. 7 Marked-images of Baboon under various payloads. **a** ER = 3/7 bpp. **b** ER = 2 bpp. **c** ER = 3 bpp

demonstrate that the security performance of the proposed scheme is better than the extended "Matrix Encoding" and the extended "Hamming+1" scheme.

Conclusions

Based on (7, 4) Hamming code, a novel high capacity data hiding scheme is proposed. Cover pixels are matched adaptively to embed data according to different embedding payloads. Compared to the related works, the image quality under high payload gets

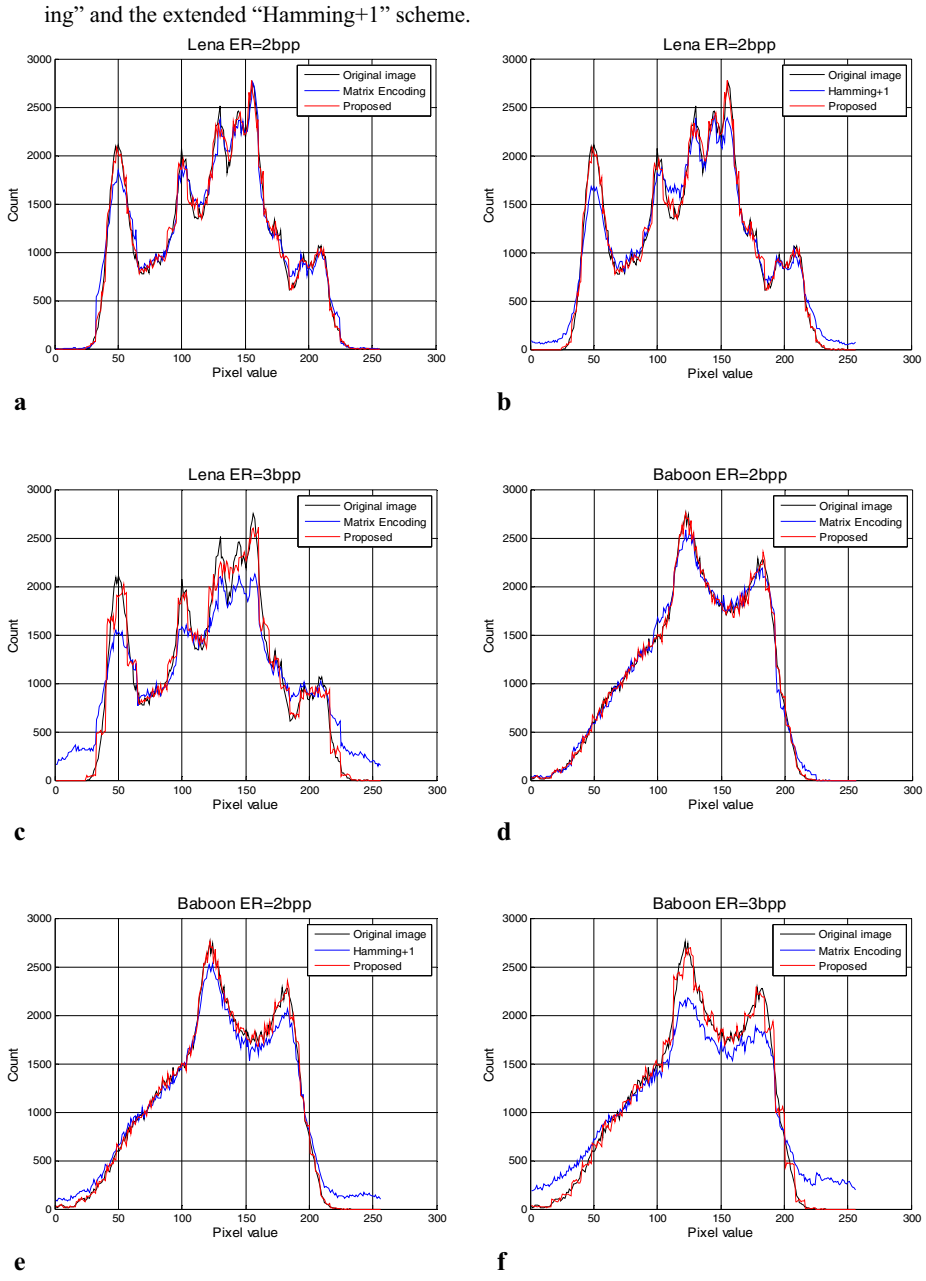

Fig. 8 The pixel histogram analysis comparison of Lena and Baboon. **a** "Matrix Encoding" of Lena with ER = 2 bpp. **b** "Hamming+1" of Lena with ER = 2 bpp. **c** "Matrix Encoding" of Lena with ER = 3 bpp. **d** "Matrix Encoding" of Baboon with ER = 2. **e** "Hamming+1" of Baboon with ER = 2 bpp. **f** "Matrix Encoding" of Baboon with ER = 3

improved significantly while maintaining visual quality under low payload. Because of the use of pixel matching, a seed can be also used to match pixels to improve the security. Moreover, this method is not limited to grayscale images, but can be also applied to color images, compressed images, audios, videos and other digital media. Future works

include investigating this scheme on other error correcting codes and improving the data embedding efficiency further.

Authors' contributions
All authors work in our laboratory and they have contributed to this work. All authors read and approved the final manuscript.

Author details
[1] Key Laboratory of Intelligent Computing & Signal Processing, School of Computer Science & Technology, Anhui University, Hefei 230601, Anhui, China. [2] School of Communication and Information Engineering, Shanghai University, Shanghai 200072, P.R. China.

Acknowledgements
This research work is supported by National Natural Science Foundation of China (61502009, 61472001), Anhui Provincial Natural Science Foundation (1508085SQF216), Project gxyqZD2016011 supported by the Key Program for Excellent Young Talents in Colleges and Universities of Anhui Province, Quality Engineering Program for Colleges and Universities in Anhui Province (2015jyxm042) and Undergraduates Training Foundation of Anhui University (J18520229, J18511158, J18515316). The test images used in this paper are all open standard test images available as TIFF files from the University of Southern California's Signal and Image Processing Institute: http://sipi.usc.edu/database/.

Competing interests
The authors declare that they have no competing interests.

Received: 3 October 2015 Accepted: 12 February 2016
Published online: 25 February 2016

References

Chang C, Chou Y (2008) Using nearest covering codes to embed secret information in gray scale images. In: Proceedings of the 2nd international conference on ubiquitous information management and communication. ACM, pp 315–320

Chen L, Lu L, Hu A, Sun X (2013) Improved information hiding algorithm based on twice positioning in coding channel. J Commun 34(12):120–130

Crandall R (1998) Some notes on steganography. Posted on steganography mailing list. http://os.inf.tudresden.de/westfeld/Crandall.pdf

Feng G, Lan Y, Zhang X, Qian Z (2015) Dynamic adjustment of hidden node parameters for extreme learning machine. IEEE Trans Cybern 45(2):279–288

Hong W, Chen T, Chen J (2015) Reversible data hiding using Delaunay triangulation and selective embedment. Inf Sci 308:140–154

Ker A, Bas P, Böhme R, Cogranne R, Craver S, Filler T, Fridrich J, Pevny T (2013) Moving steganography and steganalysis from the laboratory into the real world. In: Proceedings of the First ACM workshop on information hiding and multimedia security. ACM, pp 45–58

Liu C (2007) Research on theory and application of steganography based on error-correcting code. Dissertation, the PLA Information Engineering University

Ma Z, Li F, Zhang X (2013) Data hiding in halftone images based on hamming code and slave pixels. J Shanghai Univ (Nat Sci) 19(2):111–115

Qian Z, Zhang X (2015) Reversible data hiding in encrypted image with distributed source encoding. IEEE Trans Circuits Syst Video. doi:10.1109/TCSVT.2015.2418611

Wang X (2009) Research on channel coding based information hiding techniques. Dissertation, Harbin Institute of Technology

Xia Z, Wang X, Sun X, Wang B (2014a) Steganalysis of least significant bit matching using multi-order differences. Secur Commun Netw 7(8):1283–1291

Xia Z, Wang X, Sun X, Liu Q, Xiong N (2014) Steganalysis of LSB matching using differences between nonadjacent pixels. Multimed Tools Appl 1–16. doi:10.1007/s11042-014-2381-8

Yin Z, Chang C, Zhang Y (2010) An information hiding scheme based on (7, 4) hamming code oriented wet paper codes. IJICIC 6(7):3121–3130

Zhang W, Wang S, Zhang X (2007) Improving embedding efficiency of covering codes for applications in steganography. IEEE Commun Lett 11(8):680–682

Zhu X, Liu J, Zhang W (2010) A steganographic algorithm based on hamming code and wet paper code. J Electron Inf Technol 32(1):162–165

Zielińska E, Mazurczyk W, Szczypiorski K (2014) Trends in steganography. Commun ACM 57(3):86–95