Adaptive Data Hiding in Edge Areas of Images With Spatial LSB Domain Systems

Cheng-Hsing Yang, Chi-Yao Weng, Shiuh-Jeng Wang, Member, IEEE, and Hung-Min Sun

IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 3, NO. 3, SEPTEMBER 2008

Prof. J. K. Mandal,
Dean Faculty of Engineering, Technology & Management
University of Kalyani,
Kalyani, West Bengal, India, Mail: jkmandal@klyuniv.ac.in
Abstract

This paper proposes a new adaptive least-significantbit (LSB) Steganographic method using pixel-value differencing (PVD) that provides a larger embedding capacity and imperceptible stegoimages. The method exploits the difference value of two consecutive pixels to estimate how many secret bits will be embedded into the two pixels. Pixels located in the edge areas are embedded by a -bit LSB substitution method with a larger value of than that of the pixels located in smooth areas. The range of difference values is adaptively divided into lower level, middle level, and higher level. For any pair of consecutive pixels, both pixels are embedded by the 3/4/5-bit LSB substitution method. However, the value is adaptive and is decided by the level which the difference value belongs to. In order to remain at the same level where the difference value of two consecutive pixels belongs, before and after embedding, a delicate readjusting phase is used.
**Pixel Value Differencing Method**

$S_1$: Pixels are grouped in row major order taking two adjacent pixels together (say $A_{00}, A_{01}$)

<table>
<thead>
<tr>
<th>$A_{00}, A_{01}$</th>
<th>$A_{02}, A_{03}$</th>
<th>$A_{04}, A_{05}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{10}, A_{11}$</td>
<td>$A_{12}, A_{13}$</td>
<td>$A_{14}, A_{15}$</td>
</tr>
<tr>
<td>$A_{20}, A_{21}$</td>
<td>$A_{22}, A_{23}$</td>
<td>$A_{24}, A_{25}$</td>
</tr>
<tr>
<td>$A_{30}, A_{31}$</td>
<td>$A_{32}, A_{33}$</td>
<td>$A_{34}, A_{35}$</td>
</tr>
</tbody>
</table>
$S_2$: Take A00 and A01 as Pi and Pi+1; Let $A_{00}=P_i=64$ and $A_{01}=P_{i+1}=47$: Binary value is $P_i=01000000$ and $P_{i+1}=00101111$; Div=17
Let information to be embedded as “10100000”

$S_3$: Compute $d_i = |P_i - P_{i+1}| = |64 - 47| = 17$: See the interval where $d_i$ belongs to: $d_i \in \{l_i, m_i, u_i\}$ i.e. $d_i \in \{16,31\}$

$S_4$: As $d_i$ belongs to $\{16,17\}$, i.e. middle level; 4 lsb bits of both $P_i$ and $P_{i+1}$ to be replaced by 4 information bits each i.e. 0000 of $P_i$ to be replaced by 1010 and 1111 of $P_{i+1}$ to be replaced by 0000.

So $P_i = 64 = 01000000$ will be $01001010 = P_i' = 74$
$P_{i+1} = 47=00101111$ will be $00100000=P_{i+1}'=32$
$P_i'$ and $P_{i+1}'$ are Stego Pixels
Status of Pixels

\[ P_i = 64 \]
\[ P_i' = 74 \]
\[ P_{i+1} = 47 \]
\[ P_{i+1}' = 32 \]

Increased by 10 in 4-bit insertion

\[ d_i = |P_i - P_{i+1}| = |64 - 47| = 17 \]

See the interval where \( d_i \) belongs to: \( d_i \in \{l_i, m_i, u_i\} \) i.e. \( d_i \in \{16, 31\} \)-middle range

<table>
<thead>
<tr>
<th>Lower Range</th>
<th>Middle Range</th>
<th>High Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>16-31</td>
<td>32-255</td>
</tr>
<tr>
<td>l</td>
<td>m</td>
<td>h</td>
</tr>
<tr>
<td>3-bit hiding</td>
<td>4-bit hiding</td>
<td>5-bit hiding</td>
</tr>
</tbody>
</table>

\[ P_i' = 74 \]
\[ P_{i+1}' = 32 \]
**Formula for Readjustment**

**Mendatory**

- Pixels $P_i'$ and $P_{i+1}'$ are readjusted as follows

\[
(P_i', P_{i+1}') = \begin{cases} 
P_i' - 8/16/32, & P_{i+1}' + 8/16/32 \quad \text{if } P_i' \leq P_{i+1}' \quad \ldots\ldots(1) \\
P_i + 8/16/32, & P_{i+1} - 8/16/32 \quad \text{if } P_i' < P_{i+1}' \quad \ldots\ldots(2)
\end{cases}
\]

Here 4 lsb bits are replaced in $P_i$ and $P_{i+1}$ the unaltered bit is fifth bit, hence $2^4 = 16$ to be added/subtracted as adjustment

- Present case: $P_i' = 74 \geq P_{i+1}' = 32$, hence Eq. (1) hold and

\[
P_i' = P_i' - 16 = 74 - 16 = 58 \\
P_{i+1}' = P_{i+1}' + 16 = 32 + 8 = 48
\]

Hence after embedding pixel pair $(A_{00}, A_{01}) = (64, 47)$ will be $(58, 48)$ with information “10100000” where “1010” is embedded in 4-lsb of $P_i$ and “0000” is embedded in 4-lsb of $P_{i+1}$
Correctness of Adjustment

The logic behind adding/subtracting 16 with two adjacent pixels:
If the range is shifted from lower to higher the embedded message become undetectable. To adjust pixels 16 is added or subtracted for four bit hiding to bring the interval in the lower range. Again if you add or subtract 16 from $P_i'$ or $P_{i+1}'$ then 4-lsb bits in both cases will be unaltered as there will be change on 4\textsuperscript{th} bit position(from LSB(0\textsuperscript{th} bit)) towards MSB.

$P_i$ after adjustment = 58 = 0011 1010

This bit has changed during handle

Unchanged Embedded bits
Correctness of Adjustment

- After embedding and before readjustment, the pixel $P_i'$ was $74_{10} = 01001010$
  - Adjustment $-16_{10} = 00010000$
  - $P'_i$ on readjustment $58 = 00111010$

- After embedding and before readjustment, the pixel $P_{i+1}'$ was $32_{10} = 00100000$
  - Adjustment $+16_{10} = 00010000$
  - $P'_{i+1}$ on readjustment $48 = 00110000$

This bit is changed in handling.

Embedded information

No effect on information embedded.

This bit changed in readjustment, no effect on information.
Calculate new range \( d_i' = 58 - 48 = 10 \)

The interval/range fabricated to lower range after adjustment [Step 4 of the algorithm]

<table>
<thead>
<tr>
<th>Pixels</th>
<th>Value before embedding</th>
<th>Value after embedding</th>
<th>Value after readjustment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_i )</td>
<td>64</td>
<td>74</td>
<td>58</td>
</tr>
<tr>
<td>( P_{i+1} )</td>
<td>47</td>
<td>32</td>
<td>48</td>
</tr>
</tbody>
</table>
Hence after modified LSB Substitution (step 4) as a result new $P_i' = 58$ and $P_{i+1}' = 48$ and $d_i' = |58 - 48| = 10$ Where $d_i = 17$.

**Better Choice (clause 6)**

$d_i \in \text{Middle Level} \& d_i' \in \text{Lower Level}$ and $P_i' = 58 > P_{i+1}' = 48$ better choice will be between (Step 6, case 6.2))

$(P_i' + 2^k, P_{i+1}')$ and $(P_i', P_{i+1}' - 2^k)$
Handle

=> (58 + 2^4, 48) and (58, 48 - 2^4)

=> (58 + 16, 48) and (58, 48 - 16)

=> (74, 48) and (58, 32)

Calculate the difference/deviation

\[
\begin{array}{c|c|c}
74,48 & 58,32 \\
64,47 & 64,47 \\
10,01 & 06,15 \\
(10)^2 + (01)^2 = 101 & (06)^2 + (15)^2 = 261
\end{array}
\]
Dev. First Set

\((10)^2 + (01)^2 = 101\)

Dev. Second Set

\((06)^2 + (15)^2 = 261\)

As \(101 < 261\), Deviation in the first set is minimum. Hence first set i.e., 74, 48 is better choice

\[ P_i', P_{i+1}' = 74, 48 \]

And \(d_i = |P_i' - P_{i+1}'| = |74 - 48| = 26\) belongs to middle level
Hence two pixels after embedding \(S = 10100000\) is 74 and 48
Decoding

Same procedure is followed as encoding

$$A_{oo} = P_i = 74 \quad A_{oi} = P_{i+1} = 48$$

Calculate new range $$d_i = |74-48| = 26$$

The interval/range fabricated out of $$l(3)-m(4)-h(5)$$

$$\{[0-15],[16-31],[32,255]\}$$ is $$[16-31]$$ i.e. middle range i.e., $$d_i \in \{16,31\}$$

Hence 4 bits from LSB are embedded in 4 lsbs of $$P_i'$$ and $$P_{i+1}'$$ and $$P_i' = 01001010$$ and $$P_{i+1}' = 00110000$$

- 4-bits from lsb of $$P_i' = 1010$$
- 4-bits from lsb of $$P_{i+1} = 0000$$
- Concatenating these two string we obtain $$10100000 = S$$
Pixels are grouped into row major order taking two adjacent pixel together (say $A_{00}, A_{01}$)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{00}, A_{01}$</td>
<td>$A_{02}, A_{03}$</td>
<td>$A_{04}, A_{05}$</td>
</tr>
<tr>
<td>$A_{10}, A_{11}$</td>
<td>$A_{12}, A_{13}$</td>
<td>$A_{14}, A_{15}$</td>
</tr>
<tr>
<td>$A_{20}, A_{21}$</td>
<td>$A_{22}, A_{23}$</td>
<td>$A_{24}, A_{25}$</td>
</tr>
<tr>
<td>$A_{30}, A_{31}$</td>
<td>$A_{32}, A_{33}$</td>
<td>$A_{34}, A_{35}$</td>
</tr>
</tbody>
</table>
\[ S_2: \text{Take } A_{00} \text{ and } A_{01} \text{ as } P_i' \text{ and } P_{i+1}'; \text{ Let } A_{00} = P_i' = 74 \text{ and } A_{01} = P_{i+1}' = 48: \text{ Binary value is } P_i = 01001010 \text{ and } P_{i+1} = 01000000; \text{ Div} = 26 \]

Information embedded was \( S = "10100000" \)

\[ S_3: \text{Compute } d_i = |P_i' - P_{i+1}'| = |74 - 48| = 26: \text{ See the interval where } d_i \text{ belongs to: } d_i \in \{l_i, m_i, u_i\} \text{ i.e. } d_i \in \{16, 31\} \]

\[ S_4: \text{As } d_i \text{ belongs to } \{16, 31\}, \text{ i.e. lmiddle level 4 lsb bits of both } P_i ' \text{ and } P_{i+1} ' \text{ to be strip off as 4 information bits each i.e. 1010 from } P_i' \text{ and 0000 from } P_{i+1} ' \text{ to obtain hidden message.} \]

4-bits from lsb of \( p_i = 1010 \)
4-bits from lsb of \( p_{i+1} = 0000 \)
Concatenating these two string we obtain 10100000

So extracted \( S = 10100000 \)
Two cover images. (a) Elaine. (b) Baboon
Two stegoimages created by our approach with 3-4 division (a) Elaine (embedded data are 883,196 bits, PSNR is 33.52 dB). (b) Baboon (embedded data are 916,010 bits, PSNR is 33.01 dB).
Two stegoimages created by our approach with 3-4-5 division (a) Elaine (embedded data are 816 956 bits, PSNR is 37.82dB). (b) Baboon (embedded data are 874 642 bits, PSNR is 34.84 dB).
### Results of Capacities and PSNRS Using various L-H Divisions in Cheng-Hsing Yang, Chi-Yao Weng, Shiuh-Jeng Wang method

<table>
<thead>
<tr>
<th>Cover image</th>
<th>1-h Capacity</th>
<th>2-3 PSNR</th>
<th>2-4 Capacity</th>
<th>2-4 PSNR</th>
<th>3-4 Capacity</th>
<th>3-4 PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elaine</td>
<td>621052</td>
<td>42.57</td>
<td>717816</td>
<td>34.17</td>
<td>883196</td>
<td>33.52</td>
</tr>
<tr>
<td>Lena</td>
<td>575188</td>
<td>43.95</td>
<td>626088</td>
<td>36.96</td>
<td>837332</td>
<td>36.28</td>
</tr>
<tr>
<td>Baboon</td>
<td>653866</td>
<td>42.08</td>
<td>783444</td>
<td>34.2</td>
<td>916010</td>
<td>33.01</td>
</tr>
<tr>
<td>Pepper</td>
<td>561236</td>
<td>44.49</td>
<td>598184</td>
<td>38.11</td>
<td>823380</td>
<td>37.17</td>
</tr>
<tr>
<td>Toys</td>
<td>565679</td>
<td>44.29</td>
<td>607068</td>
<td>38.07</td>
<td>827822</td>
<td>37.37</td>
</tr>
<tr>
<td>GIRL</td>
<td>579686</td>
<td>43.92</td>
<td>635084</td>
<td>36.54</td>
<td>841830</td>
<td>36.39</td>
</tr>
<tr>
<td>Gold</td>
<td>581674</td>
<td>43.72</td>
<td>639060</td>
<td>36.14</td>
<td>843818</td>
<td>35.79</td>
</tr>
<tr>
<td>Barb</td>
<td>629976</td>
<td>42.75</td>
<td>735664</td>
<td>35.50</td>
<td>892120</td>
<td>34.93</td>
</tr>
<tr>
<td>Zelda</td>
<td>556840</td>
<td>44.58</td>
<td>589392</td>
<td>37.91</td>
<td>818984</td>
<td>36.62</td>
</tr>
<tr>
<td>Tiffany</td>
<td>566992</td>
<td>44.23</td>
<td>609696</td>
<td>37.52</td>
<td>829136</td>
<td>37.10</td>
</tr>
<tr>
<td>Average</td>
<td>589219</td>
<td>43.66</td>
<td>654150</td>
<td>36.51</td>
<td>851363</td>
<td>35.81</td>
</tr>
</tbody>
</table>
**Restoration of Pixels after extraction**

- After extraction replace 4-lsb values of each pixels with ‘0000’

**Embedded Pixels**  

<table>
<thead>
<tr>
<th>Pixels</th>
<th>Original Value</th>
<th>Value after embedding</th>
<th>Value after LSB-adjustment</th>
<th>Value on Choice</th>
<th>Value after Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_i$</td>
<td>64</td>
<td>74</td>
<td>58</td>
<td>74</td>
<td>64</td>
</tr>
<tr>
<td>$P_{i+1}$</td>
<td>47</td>
<td>32</td>
<td>48</td>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>
## Restoration of Pixels after extraction

<table>
<thead>
<tr>
<th>Pixels</th>
<th>Original</th>
<th>Value after Extraction</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_i$</td>
<td>64</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>$P_{i+1}$</td>
<td>47</td>
<td>48</td>
<td>1</td>
</tr>
</tbody>
</table>

Deviation is Minimum
Features

1. 3-bit, 4-bit and 5-bit are replaced from LSB directly without any hash function, containing three boundaries [0-15], [16-31], [32-255]

2. Immediately after embedding Modification of LSB (Modified LSB) must be done, where three handles are used as follows
Features

Handle I

If $d_i \in [0-15]$, lower level then $l=3$ and $2^l$ i.e., $2^3$ is added to $P_i$ ‘ if $P_i'<P_i$ else $2^l$ i.e., $2^3$ is subtracted from $P_i$

Handle II

If $d_i \in [16-31]$, middle level then $m=4$ and $2^m$ i.e., $2^4$ is added to $P_i$ ‘ if $P_i'<P_i$ else $2^m$ i.e., $2^4$ is subtracted from $P_i$

Handle III

If $d_i \in [32-255]$, higher level then $m=5$ and $2^h$ i.e., $2^5$ is added to $P_i$ ‘ if $P_i'<P_i$ else $2^h$ i.e., $2^5$ is subtracted from $P_i$
3. Adjustment phase is executed when new di’ is calculated (New difference after handle is done) and adjustment algorithm is executed depending on new di’ to readjust and bring the interval where it belongs in the input image.
Thank You