

Zero-distortion Authentication Watermarking

Yongdong Wu

Institute for Infocomm Research, Singapore

Content



- LAW(Lossless Authentication Watermarking)
- ZAW(Zero-distortion Authentication Watermarking)
- ZAW Application on Palette Images Authentication
- ZAW Application on Software Authentication
- Conclusion

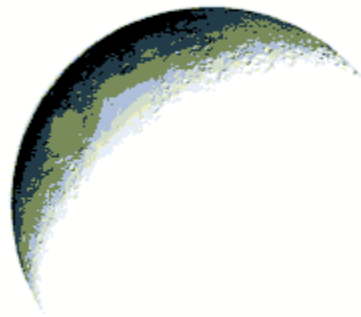
LAW

- Select pixels or coefficients
- Compress them so as to save space for the watermark.

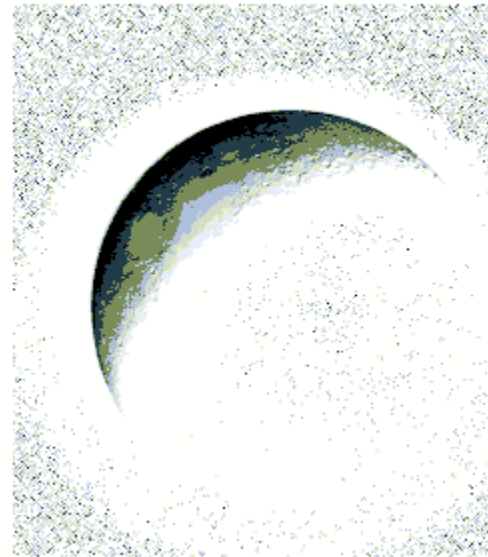
Advantage: the embedding distortion can be completely removed from the watermarked image.

Disadvantage: The image will be distorted if a dedicated decoder for signal recovery is not involved.

cont

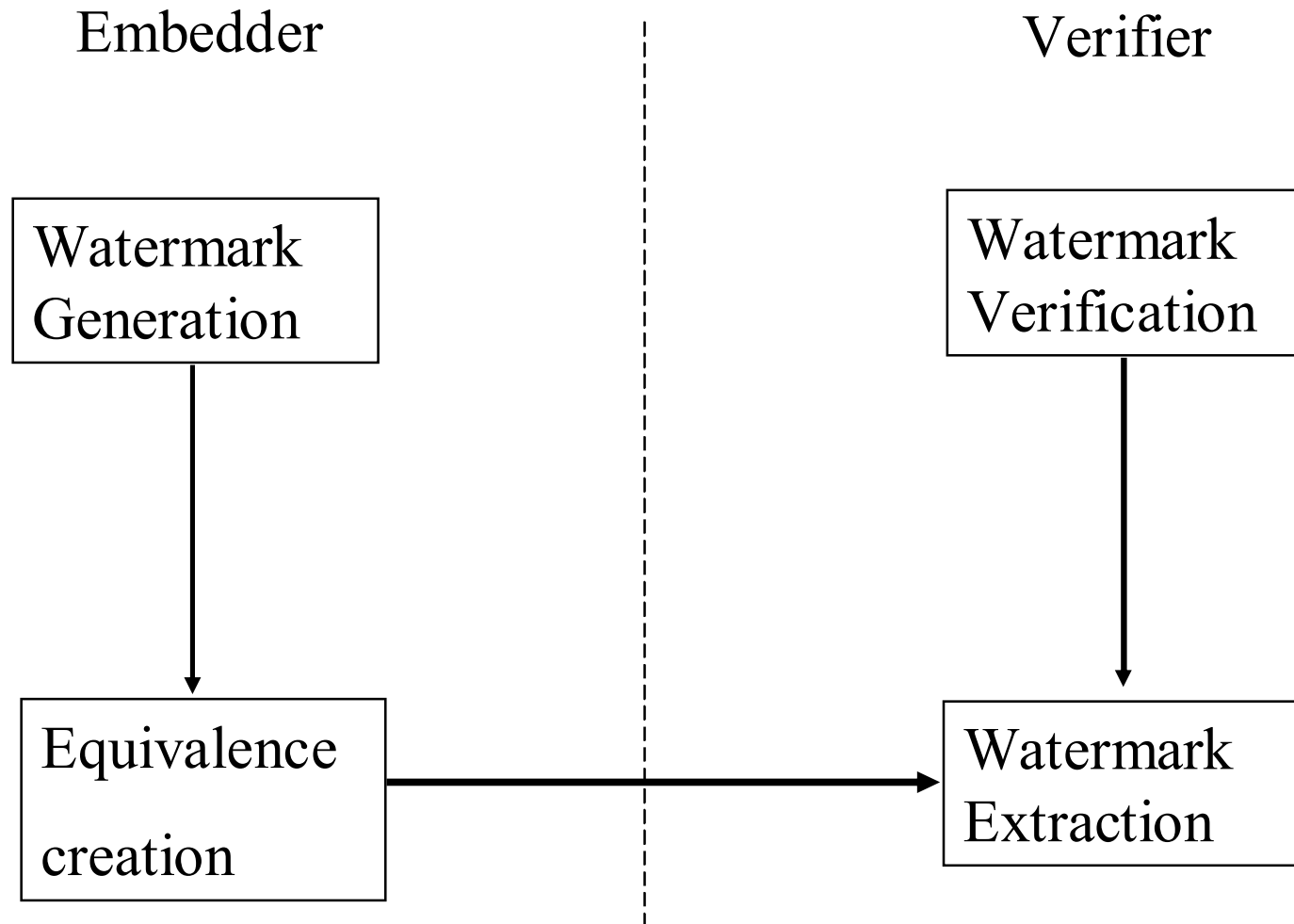


Original Image



Watermarked Image
without Recovery

Framework of ZAW



Palette Image

A palette image (e.g. GIF) includes at least two parts.

- **Palette**: maps color indexes to their corresponding components.
- **Image data** : represent each pixel with the color index rather than the color components.

Table 1a: Palette

Color	Red	Green	Blue
0	255	255	255
1	255	0	0
2	0	255	0
3	0	0	255

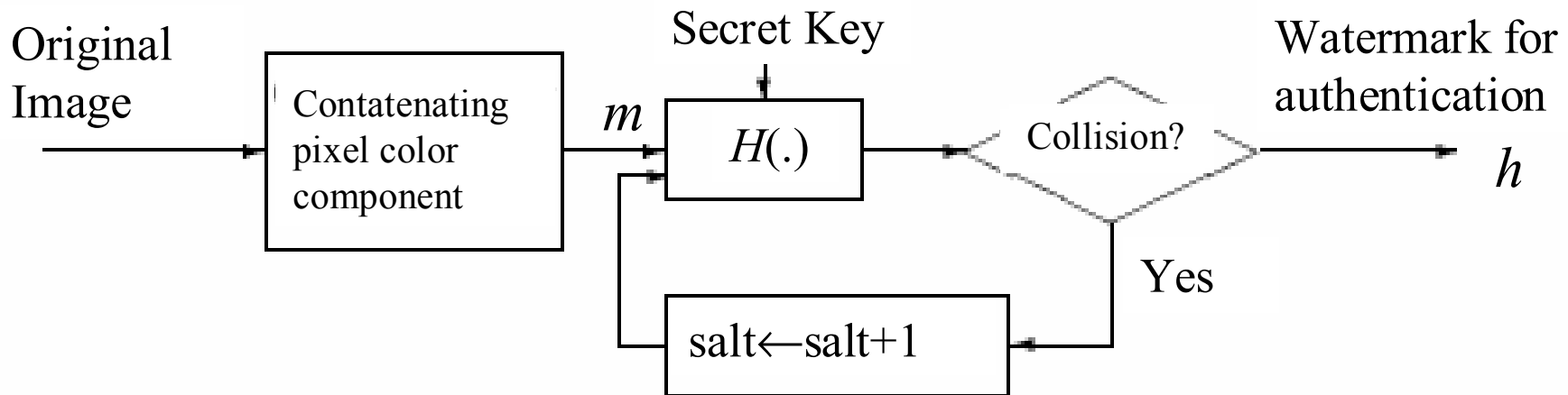
Table 1b: Image data

0	0	0	2	3
2	2	2	1	3
1	1	1	1	3
3	3	3	3	3

Table 1c: Pixels

W	W	W	G	B
G	G	G	R	B
R	R	R	R	B
B	B	B	B	B

LAW on Palette Image: Watermark Generating



$$m = R_1 \| G_1 \| B_1 \dots R_k \| G_k \| B_k \dots R_n \| G_n \| B_n$$

R_k, G_k, B_k are the red, green and blue values of k^{th} pixel.

Watermark Embedding: Color Numbering



- Define a natural palette whose entries are sorted by $rgb = 65536R + 256G + B$. (R, G, B) is a triple of red, green, and blue values.
- To embed watermark $h = h_0 h_1, \dots, h_t$, move entry h_k ($k = 0, 1, \dots, t$) in the natural palette to entry k in the new palette.
- Regulate the rest $C-t-1$ entries to embed some other messages
- Change the image data with the new palette such that the components of each pixel are unchanged.

Example

Original

Table 1a: Palette

Color	Red	Green	Blue
0	255	255	255
1	255	0	0
2	0	255	0
3	0	0	255

Table 1b: Image data

0	0	0	2	3
2	2	2	1	3
1	1	1	1	3
3	3	3	3	3

Table 1c: Pixels

W	W	W	G	B
G	G	G	R	B
R	R	R	R	B
B	B	B	B	B

Watermarked

Table 2a: Palette

Color	Red	Green	Blue
0	0	0	255
1	0	255	0
2	255	0	0
3	255	255	255

Table 2b: Image data

3	3	3	1	0
1	1	1	2	0
2	2	2	2	0
0	0	0	0	0

Table 2c: Pixels

W	W	W	G	B
G	G	G	R	B
R	R	R	R	B
B	B	B	B	B

Watermark Embedding: Occurrence Numbering



- Calculate the occurrence of each color in the image data
- Sort the occurrence in decreasing order.
- In the new palette, let color h_0 to be the color with highest occurrence, h_1 to be the color with the second highest occurrence, and so on. If there are two colors with the same occurrence frequency, the smaller color index in natural palette is firstly assigned.
- Re-order the rest of palette entries.
- Change the image data with the new palette such that the components of each pixel are unchanged.

Example

Original

Table 1a: Palette

Color	Red	Green	Blue
0	255	255	255
1	255	0	0
2	0	255	0
3	0	0	255

Table 1b: Image data

0	0	0	2	3
2	2	2	1	3
1	1	1	1	3
3	3	3	3	3

Table 1c: Pixels

W	W	W	G	B
G	G	G	R	B
R	R	R	R	B
B	B	B	B	B

Watermarked

Table 3a: Palette

Color	Red	Green	Blue
0	0	255	0
1	255	255	255
2	255	0	0
3	0	0	255

Table 3b: image data

1	1	1	0	3
0	0	0	2	3
2	2	2	2	3
3	3	3	3	3

Table 3c: Pixels

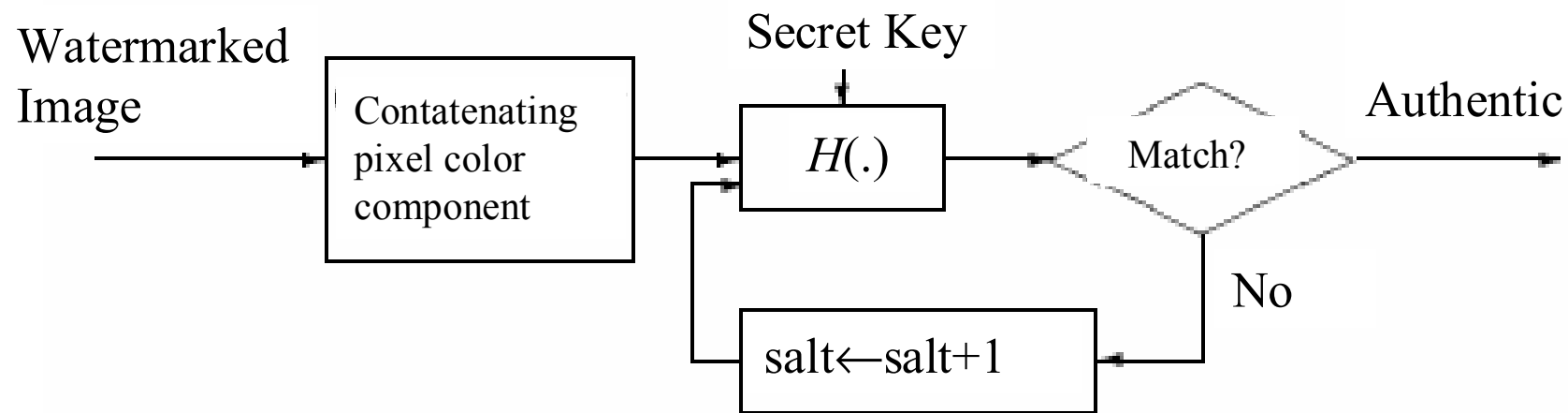
W	W	W	G	B
G	G	G	R	B
R	R	R	R	B
B	B	B	B	B

Watermark Extraction: Occurrence Numbering



- Calculates the occurrence of the image colors.
- Sorts the occurrence in decreasing order.
- The color indexes of palette corresponding to the highest occurrences are the watermark.

LAW on Palette Image: Watermark Verifying



LAW on software: Embedding



Assume : a program includes independent modules f_0, f_1, \dots, f_m .
i.e., for any $i \neq j$, there are no control flow and data flow between any pair of f_i and f_j .

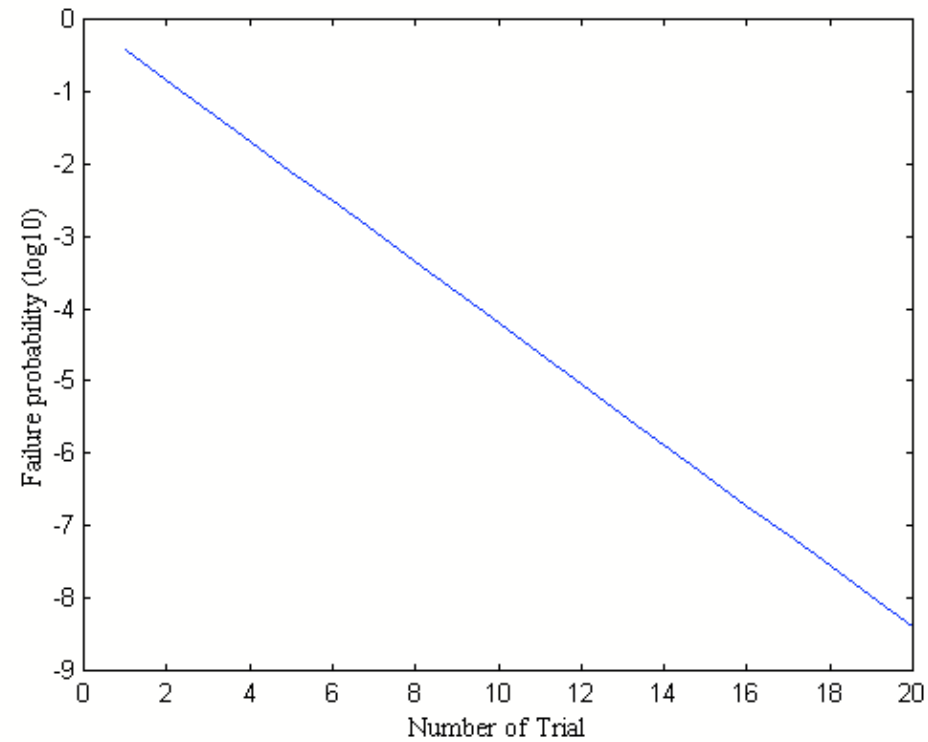
- The hash values of the modules are F_0, F_1, \dots, F_m . The MAC for this program is $h = H(K || F_0 || F_1 || \dots || F_m)$
- Let $j = p(i, h)$, $i = 0, 1, \dots, m$. $p(\cdot)$ is a one-way permutation.
- Re-order the modules of the original software into f_0', f_1', \dots, f_m' to generate another software, where $f_j' = f_i$, $j = 0, 1, \dots, m$.

LAW on software: Verifying



- Calculate the hash value $F_0', F_1' < \dots < F_m'$ for all modules.
- Sort the hash values by increasing order values $F_0'' < F_1'' < \dots < F_m''$
- Calculate the $h' = H(K || F_0'' || F_1'' || \dots || F_m'')$.
- For all $i = 0, 1, \dots, m$, if $F_j' = F_i''$ where $j = p(i, h')$, the software is authentic, otherwise, it is modified.

Number of Trial



256-color GIF file. Trial 16 times failure probability is 1.9×10^{-7}

Conclusion



- ZAW creates the equivalence of the original signal. No recovery procedure is desired at the detector side.
- ZAW application on palette images by exploiting color entries in the palette
- ZAW application on software authentication by embedding the watermark into the software instruction orders.
- ZAW is applicable to some file formats only.

Thanks