LETTER

A New Binary Image Authentication Scheme with Small Distortion and Low False Negative Rates*

Younho LEE[†a)], **Junbeom HUR**[†], **Heeyoul KIM**[†], ***Nonmembers***, **Yongsu PARK**[††b)], ***and*** **Hyunsoo YOON**[†], ***Members***

SUMMARY In this study, a novel binary image authentication scheme is proposed, which can be used to detect any alteration of the host image. In the proposed scheme, the watermark is embedded into a host image using a Hamming-code-based embedding algorithm. A performance analysis shows that the proposed scheme achieves both smaller distortion and lower false negative rates than the previous schemes.

key words: *image authentication, binary image, fragile watermarking, multimedia security*

1. Introduction

Binary images offer many benefits in various fields such as faxes or digitally-processed signature. However, it can be easily forged during an image communication.

To cope with the problem of binary image forgery in digital media, several image authentication schemes were proposed [1]–[3], [5], [8]. Image authentication methods aim to detect any change of a host image by embedding integrity-check code such as watermarks. The previous image authentication schemes, however, have relatively high false negative rates (the probability that the altered image can pass the verification test) [2], [8]; or distort a relatively large number of pixels in the host image to embed the watermark [1], [3], [5]. This not only degrades the quality of the host image, but also gives an adversary a good chance to detect the presence of the watermarks via statistical tests.

In this letter, a novel authentication scheme for binary images is proposed. In this scheme, $n \cdot \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit watermark can be embedded into the c-bit host image with n blocks by flipping only n pixels. For each block, the Hamming-code-based (HCB) embedding technique [6], [7] is employed to embed the watermark chunk by flipping only one pixel. Then, a single additional pixel in each block is flipped to conceal the presence of the watermark chunk. To select the position of this flipped pixel, we use all the context of the remaining $n-1$ blocks, which significantly reduces the false negative rate of the proposed scheme. Later at the watermark extraction and verification phase, these additionally flipped pixels are recovered to have the original value.

The organization of this letter is as follows. A new binary image authentication scheme is proposed in Sect. 2. Performance comparison between the proposed scheme and previous works is provided in Sect. 3. Finally, the conclusion is given in Sect. 4.

2. Proposed Scheme

The proposed scheme consists of the following two algorithms. Given a *host image*, the *Stego-image generation algorithm* embeds a watermark to generate the *stego-image*. From the stego-image, *Stego-image verification algorithm* can reconstruct the watermark. The validity of the stego-image can be checked by intactness of the extracted watermark.

Assume that there is a c-bit host image i, which can be divided into n blocks of $\lfloor c/n \rfloor$ bits. It is also assumed that the Stego-image generation algorithm and the Stego-image verification algorithm share a secret key $K \in \{0,1\}^t$ (t is a security parameter), a pseudo-random function $f_K : \{0,1\}^* \rightarrow \{0,1\}^{\lfloor c/n \rfloor}$ (K is a seed), and the secure hash function $H : \{0,1\}^* \rightarrow \{0,1\}^t$.

2.1 Stego-Image Generation Algorithm

For the c-bit host image i, this algorithm can embed the $n \cdot \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit watermark as follows. First, for each $\lfloor c/n \rfloor$-bit block in i, the HCB watermark embedding algorithm [6], [7] is employed to embed each $\lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit piece of the watermark by flipping one pixel. Detailed description of HCB is in Appendix.

Let the produced n intermediate stego-blocks be $s_1, \ldots, s_n$. With these blocks, the watermark concealing algorithm, **AuthHide**, outputs the final stego-image. **AuthHide** flips a single pixel of each block to conceal the watermark. The position of the flipped pixel is determined by

Manuscript received March 12, 2007.
Manuscript revised June 12, 2007.

[†]The authors are with CS Division at KAIST, Korea.
[††]*Corresponding author*. The author is with the College of Information and Communications at Hanyang Univ., Korea.

*This work was supported by the Ministry of Science and Technology(MOST)/Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc) and the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2006-C1090-0603-0015). Also, this work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2006-331-D00556). Also, this work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-042-D00294).

a) E-mail: yhlee@nslab.kaist.ac.kr
b) E-mail: yongsu@hanyang.ac.kr

DOI: 10.1093/ietcom/e90–b.11.3259

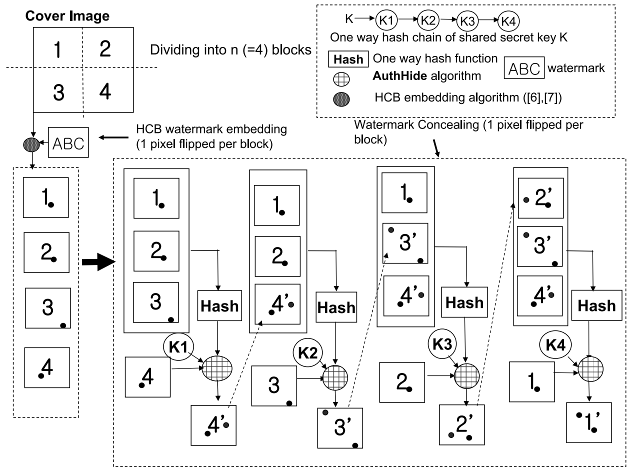
Cover Image
Dividing into n (=4) blocks
K→K1→K2→K3→K4
One way hash chain of shared secret key K
Hash One way hash function
AuthHide algorithm
ABC watermark
HCB embedding algorithm ([6],[7])
HCB watermark embedding (1 pixel flipped per block)
Watermark Concealing (1 pixel flipped per block)
Hash
K1
K2
K3
K4

Fig. 1 An overview of Stego-image generation algorithm.

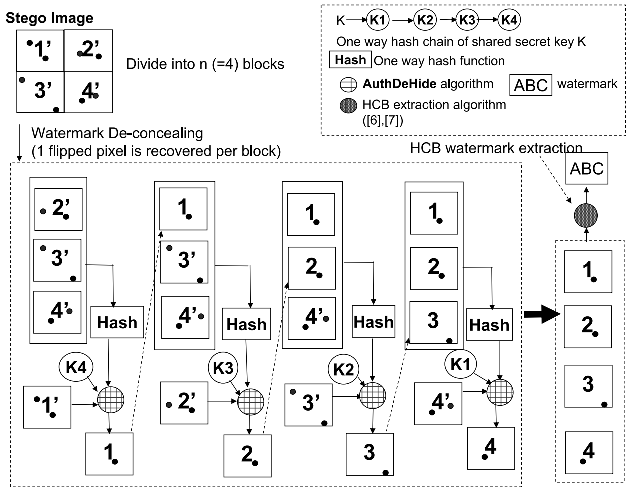
Stego Image
Divide into n (=4) blocks
K→K1→K2→K3→K4
One way hash chain of shared secret key K
Hash One way hash function
AuthDeHide algorithm
ABC watermark
HCB extraction algorithm ([6],[7])
Watermark De-concealing (1 flipped pixel is recovered per block)
HCB watermark extraction
Hash
K4
K3
K2
K1

Fig. 2 An overview of Stego-image verification algorithm.

the pseudo-random function on the inputs of the shared key K, and the hash value of concatenating all the other blocks. Fig. 1 depicts an overview of the Stego-image generation algorithm. Full description of **AuthHide** is in Algorithm 1.

2.2 Stego-Image Verification Algorithm

For the c-bit stego-image g, this algorithm can extract the $n \cdot \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit watermark as follows. For g with n blocks, the watermark de-concealing algorithm, **AuthDeHide**, produces the intermediate stego-image blocks $s_1, \ldots, s_n$; for each block, **AuthDeHide** flips the pixels that were previously flipped by **AuthHide**. The procedure of **AuthDeHide** is exactly the same as **AuthHide** except the order of the blocks to be processed: the last block that was processed by **AuthHide** is first processed by **AuthDeHide**. Fig. 2 shows an overview of the Stego-image verification algorithm. Full description of **AuthDeHide** is in Algorithm 1.

Then, for each $\lfloor c/n \rfloor$-bit s_i $(1 \leq i \leq n)$, the HCB watermark extraction algorithm [6], [7] is employed to extract each $\lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit piece of the watermark (full description is in Appendix). Finally, $n \cdot \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor$-bit watermark can be constructed by concatenating all the pieces together.

Note that for each block g_i of the stego-image g, **AuthDeHide** flips 1 pixel to produce s_i and then the extraction algorithm [6], [7] is employed to extract the piece of the watermark. If the adversary incorrectly guesses the position of the flipped pixel in g_i to produce s'_i $(\neq s_i)$, the piece of the watermark will be incorrectly extracted. Assume that the adversary (accidentally) produces s_i without K. This means that the adversary can also guess the output of the pseudo-random function f without the correct seed, which contradicts the definition of the pseudo-random function because it is assumed to be computationally infeasible to predict the output of the pseudo-random function.

In **AuthDeHide**, for each g_i of the stego-image g, all the other g_js' $(j \neq i)$ information are related to compute the flipped bit position. Hence, among n blocks, if at least one block g_i is modified, all computed s'_k $(1 \leq k \leq n)$ will be different from s_k and the extracted watermark is completely

```
/* AuthHide algorithm */
Input: $s_1, ..., s_n \in \{0,1\}^{\lfloor c/n \rfloor}$, $K$, $H(\cdot)$, $f_{\{0,1\}^t}$.
Output: $g_1, ..., g_n$.
begin
    $K_1 \leftarrow K$.
    for $i = n$ to 1 /* processing each block*/ do
        if $i = 1$ then $T = g_2 \| \cdots \| g_n$.
        else if i = n $T = s_1 \| \cdots \| s_{n-1}$.
        else $T = s_1 \| \cdots \| s_{i-1} \| g_{i+1} \| \cdots \| g_n$.
        $a \leftarrow f_{K_{n+1-i}}(H(T))$. /*compute flipped-bit position*/
        (Flip the $a$-th pixel of $s_i$)$\rightarrow g_i$.
        $K_{n+2-i} \leftarrow H(K_{n+1-i})$. /*update key for the next block*/
end
/* AuthDeHide algorithm */
Input: $g_1, ..., g_n \in \{0,1\}^{\lfloor c/n \rfloor}$, $K$, $H(\cdot)$, $f_{\{0,1\}^t}$.
Output: $s_1, ..., s_n \in \{0,1\}^{\lfloor c/n \rfloor}$.
begin
    $K_1 \leftarrow K$.
    for $i = 2$ to $n$ /* Key derivation for each block*/ do
        $K_i \leftarrow H(K_{i-1})$.
    for $i = 1$ to $n$ /*Processing each block*/ do
        if $i = 1$ then $T = g_2 \| \cdots \| g_n$.
        else if i = n $T = s_1 \| \cdots \| s_{n-1}$.
        else $T = s_1 \| \cdots \| s_{i-1} \| g_{i+1} \| \cdots \| g_n$.
        $a \leftarrow f_{K_{n+1-i}}(H(T))$. /*compute flipped-bit position*/
        (Flip the $a$-th pixel of $g_i$)$\rightarrow s_i$.
end
```

Algorithm 1: **AuthHide** and **AuthDeHide** algorithms.

crushed.

3. Performance Analysis

The performance of the proposed scheme is compared with those of [1]–[3], [5], [8]. The following two measurement metrics are used.

- *False negative rate* is a probability that a different (modified) stego-image can pass the verification test.
- *Distortion* is a required number of flipped pixels to embed the watermark.

3.1 Analysis of the Proposed Scheme

First, as for distortion, in the c-bit host image $n \cdot \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor \approx n \log_2(c/n)$-bit watermark can be contained on n-bit distortion: this is because $2n$-bit are flipped for stego-image generation and then n-flipped bits are recovered during the stego-image verification.

Next, the false negative rate of the proposed scheme is considered. For c-bit binary image, the false negative rate is equal to the rate of the number of binary images ($\in \{0,1\}^c$) that pass the Stego-image verification algorithm except the original stego-image, in all c-bit binary images ($= \{0,1\}^c$).

For an intermediate stego-block $s_i \in \{0,1\}^{\lfloor c/n \rfloor}$ ($1 \le i \le n$) having watermark chunk w_i, let the set of the binary images ($\subset \{0,1\}^{\lfloor c/n \rfloor}$) that have w_i be E_{w_i}. Then, due to the property of the Hamming code, the cardinality of E_{w_i} is $2^{\lfloor c/n \rfloor - \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor}$ for $1 \le i \le n$ [7]. Hence, if we consider all n blocks, there are $(2^{\lfloor c/n \rfloor - \lfloor \log_2(\lfloor c/n \rfloor + 1) \rfloor})^n \approx (2^{c/n - \log_2(c/n)})^n$ cases that can produce the same watermark chunks (w_is) as the original intermediate stego blocks (s_is). Since the cardinality of $\{0,1\}^c$ is 2^c, the false negative rate of the proposed scheme (FNR) can be derived as follows:

$$\mathsf{FNR} = ((2^{c/n - \log_2(c/n)})^n)/2^c = (c/n)^{-n}. \qquad (1)$$

Note that the additional pixel flipping caused by **AuthHide** algorithm does not affect FNR because the number of images that can pass the Stego-image verification algorithm, $|E_{w_i}|$, is not affected by **AuthHide** algorithm; **AuthHide** is run for each block after all the intermediate stego blocks are generated. **AuthHide** algorithm aims just to protect the watermark from altering without the correct shared secret key. As mentioned in Sect. 2, it is assumed to be computationally infeasible to guess the position of additionally flipped pixel by **AuthHide**.

3.2 Performance Comparison Result

Comparison results between the proposed scheme and the previous works are summarized at Table 1. Moreover, Fig. 3 shows comparison of the false negative rate on the same distortion. It is assumed that Kim et al.'s algorithm employs the 1024-bit RSA signature algorithm, the number of flipping pixels is 512 and number of blocks divided is 256. For fair comparison, we also assume that Wu et al.'s scheme does not employ robustness enhancement techniques such as error correcting codes because all the compared schemes are fragile watermarking schemes. Fig. 3 shows that on the same distortion the proposed scheme has far less false negative rates than the other schemes.

Table 1 Performance comparison (n: # of blocks, X: bit-length of message authentication code (or digital signature), c: bit- length of host image. †: Average Hamming distance between X-bit string (a part of host image) and the embedded authentication code. ‡: For example, 2^{-80} when 1024-bit RSA signature algorithm is used [4]. §: it is assumed that the k-bit strings that the codeholders select from the host image form a uniform-distribution (where q: # of code holder, k: bit-length of codeword). $\checkmark$: The probability that two randomly selected pixels belong to the same block. Because of odd-even encoding, the original watermark chunk can be decoded even if two additional pixels are maliciously flipped in the same block.)

Schemes	Distortion	False negative rate
Tzeng et al. [2]	$n\Sigma_{i=1}^{k} i \cdot (A_i - A_{i-1})^{\S}$ $(A_l = 1 - (2^{-k}\Sigma_{j=l+1}^{k} \binom{k}{j})^q)$	$[1-((2^k-1)/2^k)^q]$ (in [2])
Kim et al. [1], [3], [5]	$X/2^{\dagger}$	Depends on used cryptographic algorithm‡
Wu et al. [8]	$n/2$	$1/n^{\checkmark}$
Proposed scheme	n	$(c/n)^{-n}$

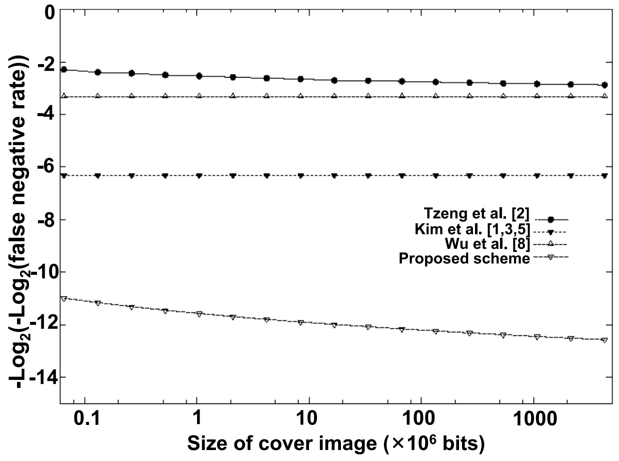

Fig. 3 False negative rate comparison when the distortion and the capacity† are 512 and 1024 bits, respectively. (†Capacity is maximum permitted bit-length of the watermark.)

4. Conclusion

In this study, a novel binary image authentication scheme was proposed. According to the analysis and comparison result, the proposed scheme has better performance in terms of distortion and false negative rate than the previous Tzeng et al.'s scheme, Wu et al.'s scheme, and Kim et al.'s scheme.

References

[1] H. Kim and A. Afif, "Secure authentication watermarking for binary images," Proc. Brazilian Symposium on Computer Graphics and Image Processing, pp.199–206, 2003.

[2] C. Tzeng and W. Tsai, "A new approach to authentication of binary images for multimedia communication with distortion reduction and security enhancement," IEEE Commun. Lett., vol.7, no.9, pp.443–445, Sept. 2003.

[3] H. Kim and A. Afif, "A secure authentication watermarking for halftone and binary images," Int. J. Imaging Syst. Technol., vol.14, no.4, pp.147–152, 2004.

[4] NIST Special Publication 800-57. (2006, May). Recommendation for Key Management—Part 1: General (Revised). [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf

[5] H. Kim, "A new public-key authentication watermarking for binary document images resistant to parity checks," Proc. IEEE International Conference on Image Processing (ICIP), vol.2, pp.1074–1077, 2005.

[6] R. Crandall, "Some notes on steganography," posted on Steganography Mailing List, 1998.

[7] A. Westfeld, "High capacity despite better steganalysis (F5-A steganographic algorithm)," Proc. Information Hiding. 4th International Workshop. LNCS, vol.2137, pp.289–302, Springer-Verlag, 2001.

[8] M. Wu and B. Liu, "Data hiding in binary images for authentication and annotation," IEEE Trans. Multimed., vol.6, no.4, pp.528–538, Aug. 2004.

[9] J. Fridrich and D. Soukal, "Matrix embedding for large payload," IEEE Trans. Information Forensics and Security, vol.1, no.3, pp.390–395, Sept. 2006.

Appendix: HCB Watermark Embedding Algorithm

The HCB watermark embedding algorithm, which is also called the matrix embedding algorithm [9], is a watermark embedding method based on the Hamming-code [6], [7]. With this algorithm, for c-bit cover image, $\lfloor \log_2(c+1) \rfloor$-bit watermark can be embedded with only at most one-bit flipping. The descriptions of the HCB watermark embedding (**HCBEmbed**) and extracting (**HCBExt**) algorithms are given in Algorithm 2.

```
/* HCBEmbed algorithm */
Input: cover image $i \in \{0,1\}^c$, watermark $m \in \{0,1\}^{\lfloor \log_2(c+1) \rfloor}$.
Output: stego image $s \in \{0,1\}^c$.
begin
    - Regard $i$ as a Hamming codeword.
    - Note that Hamming codeword $i$ consists of data bits $d$ and
    parity bits $p$. From $d$, calculate a new Hamming codeword
    $i_{ham} \in \{0,1\}^c$ where $i_{ham}$ consists of data bits $d$ and parity
    bits $p_{ham}$.
    if ($p \oplus p_{ham} \neq m$) then
        - $s \leftarrow$ the image where the bit of $(p \oplus p_{ham} \oplus m)$-th
        position in $i$ is flipped.
        ($\oplus$: bitwise XOR operation)
    else if ($p \oplus p_{ham} = m$) - $s \leftarrow i$.
    return $s$.
end
/* HCBExt algorithm */
Input: $s \in \{0,1\}^c$.
Output: $m \in \{0,1\}^{\lfloor \log_2(c+1) \rfloor}$.
begin
    - Regard $s$ as a Hamming codeword.
    - Note that $s$ consists of $(d', p')$. From $d'$, calculate a new
    Hamming codeword $i'_{ham} \in \{0,1\}^c$ where $i'_{ham}$ consists of
    $(d', p'_{ham})$.
    - return $m \leftarrow p' \oplus p'_{ham}$.
end
```

Algorithm 2: HCB watermark embedding/extracting (**HCBEmbed/HCBExt**) algorithms.