




Secure code design against collusion attacks for protecting digital content rights

Youngmo Kim¹ 

Received: 4 April 2014 / Revised: 26 June 2015 / Accepted: 4 January 2016 /

Published online: 23 January 2016

© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract This paper proposes a group-based collusion-secure code design technique for protecting copyrights on digital contents. Designing collusion-secure codes were difficult and there were problems in creating these codes in large numbers, as these codes should enable detecting collusion attacks when occurring in content with inserted forensic marks. In addition, there was a problem in applying these codes to actual services in that unique user-specific codes could not be issued. This paper proposes group anti-collusion codes (ACCs) made by introducing the concept of groups to existing balanced incomplete block design (BIBD) matrices. This would solve the problem of complexity in code design and increase the number of collusion-secure codes. For the group ACCs, problems in securing the uniqueness and verification of codes were overcome using set operations. Using the group ACCs, code complexity can be reduced when compared to existing collusion-secure codes, and code quantities can be increased when compared to code lengths.

Keywords Collusion-Secure · Forensic mark · BIBD · Balanced Incomplete Block Design (BIBD)

1 Introduction

Due to advancements in broadcast and communication, fusion technologies and rapid changes to the digital content industry, the development of distribution technologies and platforms suitable for diverse content industries accounts for a large part of digital content technology. Along with these changes, studies have been conducted on a range of topics related to broadcast communication content. For example, studies on copyright

✉ Youngmo Kim
ymkim828@ssu.ac.kr

¹ School of Computer Science and Engineering, Soongsil University, 369, Sangdo-ro, Dongjak-gu, Seoul 156-743, Korea

protection and management technologies designed to solve problems in relation to the production, distribution, and reprocessing of digital content have been intensively conducted to protect contents, and demands for those technologies have been gradually increasing in related areas [4, 5].

In particular, watermark technology can insert ownership information into digital content as well as extract information from content reproduced illegally. This technology can prove original ownership by comparing data; however, it can only identify content reproduced illegally and cannot identify distributors or their routes. On the other hand, forensic mark technology inserts information that combines owner and buyer data into the content. Therefore, the locations of forensic marks can be determined using the inserted information, which varies by content [3, 6, 12].

Collusion-secure technologies used for forensic marks ensure contents are strong against attacks that can remove and create new forensic marks used to distribute the contents illegally. These technologies include the C-Secure codes proposed by Boneh [1], the d-detecting codes proposed by Dittmann [2], and the ACC using Balanced Incomplete Block Design (BIBD) matrices, as proposed by Trappe and Wu [9]. However, these collusion-secure technologies involve design difficulties in that they cannot be created in large quantities.

Such collusion technologies that prevent illegal public usage of content after public use so that its use can be detected through reverse combination had difficulties due to issues related to the exponentially increasing length of the codes if the number of users increased. For these reasons, there were problems of being unable to produce large numbers of collusion-secure codes [7].

This paper proposes a code design algorithm for the creation of collusion-secure forensic marks to protect copyrights. The introduction presents the current state of copyright technologies followed by a discussion of collusion attacks and collusion-secure codes in Chapter 2. Chapter 3 outlines collusion-secure code designs to prevent attacks on digital contents and verification of codes. Finally, in Chapter 4, the utility of the proposed algorithm is analyzed while future study directions and expected effects are described.

2 Related studies

2.1 Collusion attack

Forensically marked contents have slightly different data values according to buyers because different buyer information is inserted into the contents. For this reason, many buyers may collude with each other to determine forensically marked positions in contents and hide the identities of conspirators by erasing forensic marks or inserting and redistributing new forensic marks using the relative differences between the contents. Attacks of this type are called collusion attacks [10].

Collusion attacks mainly use methods of reducing correlation coefficient values because forensic marks are extracted using correlation coefficients. The type of collusion attack includes averaging attacks, max/min attacks, negative-correlation attacks, zero-correlation attacks, and mosaic attacks. Figure 1 is a concept map of forensic marks and collusion attacks.

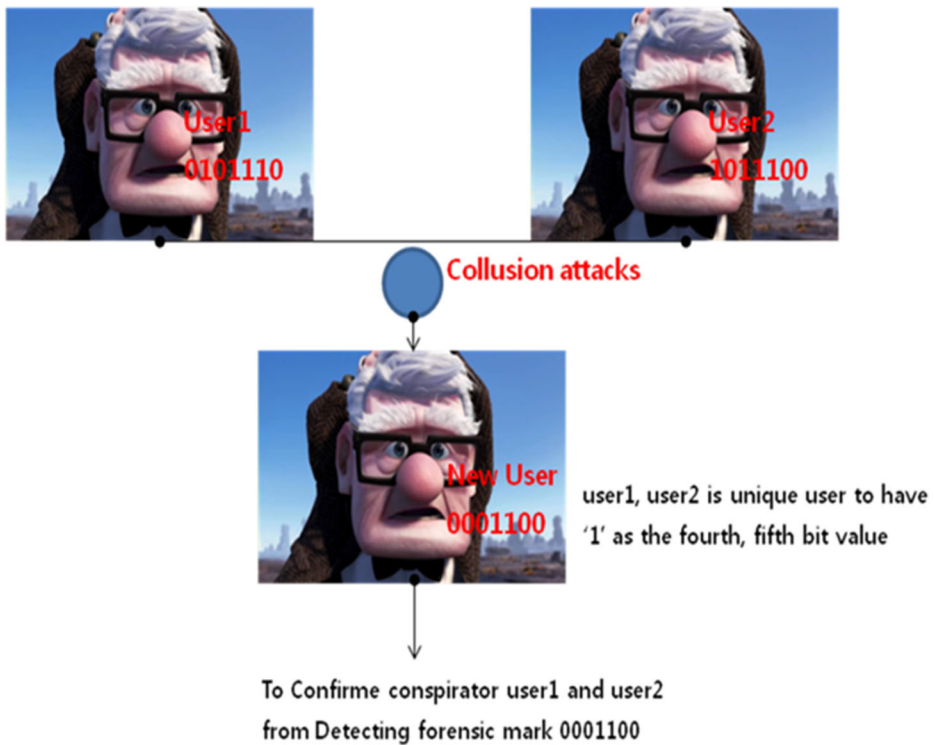


Fig. 1 System structure

2.1.1 Averaging attack

Averaging attacks are an attack technique that averages data values at the same position to create new forensically marked content.

$$\hat{e}_j = p_j + \hat{w}_j = p_j + \frac{1}{K} \sum_{i=0}^K w_{i,j} \tag{1}$$

In Expression 1, \hat{e}_j is the contents obtained through averaging attacks on K pieces of forensically marked contents. In the same expression, p_j is the coefficient value of the forensically marked contents and $w_{i,j}$ and \hat{w}_j represent the forensic mark information inserted as watermarks and those changed by an averaging attack, respectively.

The forensic mark information $w_{i,j}$ of the contents created by Expression 1 by the averaging attack includes the signals decreased by $\frac{1}{K}$.

$$c(\hat{w}, w_1) = \frac{\sum_j \left(\frac{1}{K} \sum_{i=1}^K w_{i,j} \right) \cdot w_{i,j}}{\sqrt{\left(\sum_j \left(\frac{1}{K} \sum_{i=1}^K w_{i,j} \right)^2 - \frac{1}{S} \left(\sum_j \left(\frac{1}{K} \sum_{i=1}^K w_{i,j} \right) \right)^2 \right) \left(\sum_j w_{i,j}^2 \right)}} = \frac{1}{\sqrt{K}} \tag{2}$$

Expression 2 is intended to obtain the correlation coefficient of the forensic marks of the contents colluded by an averaging attack. From Expression 2, it can be seen that the correlation coefficient value of colluded contents decreases proportionally to \sqrt{K} .

2.1.2 Max/min attack

This is an attack method proposed by Stone that obtains the minimum the maximum values from forensically marked contents that have participated in collusion. It creates new content using the average of the minimum and maximum values [8].

$$\hat{e}_j = p_j + \frac{1}{2} (\max_{i=1}^K (w_{i,j}) + \min_{i=1}^K (w_{i,j})) \tag{3}$$

In Expression 3, $\max_{i=1}^K (w_{i,j})$ indicates the maximum value among the coefficients of the K pieces of forensically marked contents that have participated in collusion and $\min_{i=1}^K (w_{i,j})$ indicates the minimum value. From Expression 4, it can be seen that the correlation coefficient of the forensic mark information $w_{i,j}$ of the content created by Expression 3 decreases proportionally to \sqrt{K} .

$$C(\hat{w}, w_i) = \sqrt{\frac{6}{(K + 1)(K + 2)}} \tag{4}$$

2.1.3 Negative-correlation attack

Negative-correlation attacks are a method proposed by Stone [8] that makes the values of correlation coefficients into negative numbers when forensic marks are extracted using correlation coefficients to make the extraction of correlation coefficients difficult. The method of making colluded contents is as shown in Expression 5 shown below.

$$\hat{e}_j = \begin{cases} p_j + \max_{i=1}^K (w_{i,j}) & \text{if } \text{med}_{i=1}^K \leq (1-\alpha)\max_{i=1}^K (w_{i,j}) + \alpha\min_{i=1}^K (w_{i,j}) \\ p_j + \min_{i=1}^K (w_{i,j}) & \text{Otherwise} \end{cases} \tag{5}$$

In Expression 5, $\max(\cdot)$, $\min(\cdot)$, and $\text{med}(\cdot)$ represent the maximum value, the minimum value, and the median value, respectively. α is a coefficient used to adjust the $\max(\cdot)$ and $\min(\cdot)$ values, and it generally has a value of 0.5. The maximum, minimum, and median values are obtained from the contents that have participated in collusion. If the average of the maximum and minimum values is smaller than the median value, the minimum value will be taken. If not, the maximum value will be taken. This will reverse the polarity of the forensic mark information to turn the value of the correlation coefficient into a negative number.

2.1.4 Zero-correlation attack

Although Stone’s negative-correlation attacks induce correlation coefficients into negative numbers, this does not mean that the forensic mark information has been erased. Zero-correlation attacks induce correlation coefficients to become close to zero so that forensic

mark information cannot be detected, as proposed by Wahadaniah et al. [10]. Expression 6 is a method of inducing correlation coefficients to become close to zero.

$$\hat{e}_j = \begin{cases} P_{j+\max_{i=1}^K(w_{i,j})} & \text{if } w_{r,j} \leq \frac{1}{2}[\max_{i=1}^K(w_{i,j}) + \min_{i=1}^K(w_{i,j})] \\ P_j + \min_{i=1}^K(w_{i,j}) & \text{Otherwise} \end{cases} \quad (6)$$

where $w_{i,j}$ represents the content that participated in collusion, and it is a targeted content. Unlike negative-correlation attacks, these attacks compare the maximum and minimum values with the forensic mark information of certain content that participated in collusion, rather than the median value, to create collusion content with a polarity opposite to that of the abovementioned content. The created collusion content is not correlated with other forensically marked content, either. That is, correlation coefficients are maintained close to zero.

2.1.5 Mosaic attack

Unlike the abovementioned attacks that make the correlation coefficient values small using the maximum and minimum values of the content that participated in collusion, this is an attack method that divides forensically marked content into many small geometric figures to create new content. Because forensic mark information is extracted as files, content divided into many pieces cannot be easily extracted.

To insert forensic mark information that is strong against mosaic attacks, the forensic mark information should be minimized, insertion areas should be in the smallest units, and the information should be inserted repeatedly so that the forensic mark information can never be removed after any form of mosaic attack.

2.2 Collusion-secure code

In general, forensic marks are made using random permutations so that those for individual buyers are not correlated with each other, providing some robustness against collusion attacks. However, there is a problem in that as the number of conspirators increases, the number of necessary codes increases exponentially. To solve this problem, the codes that have common parts at different positions by buyer are being designed and studied.

2.2.1 C-secure code

Boneh proposed c-secure codes that are robust against collusion attacks using redundancy among codes [1]. This technique defines codes to be assigned to different buyers as (l, n) -codes consisting of words where the length is l and the number is n .

$$\Gamma = \{w^{(1)}, \dots, w^{(n)} \subseteq \Sigma^l\} \quad (7)$$

In Expression 7, Σ^l represents words where the length is l , and Γ represents sets of marks that are to be inserted as forensic marks. That is, word $w^{(i)}$ is assigned to each buyer. The technique, proposed by Boneh, limited the range of collusion by presenting an insertion assumption that collusion attacks can detect inserted marks only when they are different from each other, and marks not detected cannot be changed without causing damage to the content.

Therefore, according to this assumption, the set (F) may be colluded, as shown by Expression 8.

$$F(C; \Gamma) = \left\{ w \in \left(\sum \cup \{?\} \right)^l \text{ s. t. } w|_R = w^{(u)}|_R \right\} \quad (8)$$

In this expression, C represents buyer collusion, u represents the indices of those conspirators that participated in the collusion, and R represents the positions of marks that have not been detected. $\{?\}$ represents the values of marks at positions not detected. That is, based on the insertion assumption, those sets that may be colluded include all new collusion codes that necessarily include all values that have not been detected because they overlap among codes.

To eliminate cases where collusion-created codes include the codes of buyers that did not participate in collusion, Boneh defines c-frame proof codes as those that satisfy $F(W) \cap \Gamma = W$ for all forensic marks $W \subset \Gamma$. For instance, (n, n) codes are c-frameproof codes. If $\Gamma_0(n)$ is assumed to be n -bit binary codes that have only one (1), $\Gamma_0(3)$ codes for three buyers are will be follows.

buyer1: 100
buyer2: 010
buyer3: 001

The codes that may be colluded and can be created from the three codes again become $\Gamma_0(3)$ codes, according to the definition of sets that may be colluded. Therefore, the three conspirators become unable to put the blame on other buyers using new, different codes created by collusion. Based on these assumptions, Boneh proposed a tracking algorithm that can be used to follow c-secure codes and participating conspirators. Seven codes for four buyers are as follows: where four represents the number of codes and three represents the number of repetitions.

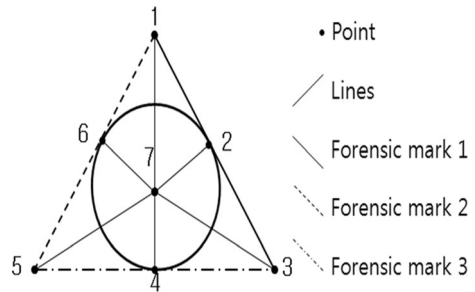
buyer1: 11111111
buyer2: 00011111
buyer3: 00000011
buyer4: 00000000

Each of the assigned codes is randomly recombined and inserted into the content. When the recombined codes have been extracted, the originally inserted codes are found through reverse combination. Once collusion has been carried out, conspirators can be tracked based on the number of one, according to positions. Although the Boneh's method is creating codes that are robust against collusion attacks in environments with limited numbers of conspirators, this method is not suitable for application in contents of limited sizes because the codes can be easily guessed. This is because they are simple and the necessary length of codes increases exponentially as the number of conspirators increases.

2.2.2 D-detecting code

As another method of creating collusion-secure codes, there are d-detecting codes proposed by Dittmann [2]. To detect two conspirators, these codes compose forensic marks based on finite projective geometry. Figure 2 shows composing forensic marks in a Pano space, which is

Fig. 2 2-detecting using Pano space



representative of an infinite projective geometry. As shown in the figure, Pano spaces are composed of points and lines. A Pano space that is composed of seven points and lines can create four forensic marks. That is, as shown in the figure, the three lines and the line that forms the circle on the center become forensic marks robust enough to protect against collusion attacks [9].

Code vectors are vectors that indicate the bit values of the positions of points that constitute lines as '1' and indicate those of other points as '0'. For instance, when the code of Forensic Mark 1 has been assigned to User1, the code vector indicates only the position bits of points 1, 2, and 3 as '1' and indicates those of the remaining four bits as '0.' That is, the code vector becomes {1110000} and this becomes the forensic mark of User 1. In the same method, the forensic mark vector of User 2 becomes {1000110} and that of User 3 becomes {0011100}. As shown in Fig. 2, forensic marks that have been composed as a Pano space always intersect at one point. Two conspirators can be tracked based on the position of this point of intersection.

Although Dittmann's 2-detecting codes show robustness against collusion attacks by two conspirators to some extent by creating commonality between codes at different positions, these codes have weaknesses in that the number of conspirators that can be detected is too limited and the codes are so simple that they can be easily assumed by attackers. In addition, the application of the code-creating method using finite projective geometry in environments with more than two conspirators is impracticable because the length of codes increases as an infinite series when the number of users as conspirators increases. In addition, composing finite projective geometry in Internet environments where many users should be supported is very difficult.

2.2.3 Anti-Collusion Code(ACC)

Trappe and Wu proposed ACCs for multimedia data using BIBD to enable detecting at least one conspirator [9, 11]. ACCs constitute code vectors that enable detecting k conspirators from the positions of bit value '1' in the bit string, which are obtained by taking the logical operator AND from the forensic mark vector $C = \{c^1, c^2, \dots, c^n\}$ of k conspirators. A basic idea is to create unique codes for individual conspirators and track the conspirators based on the positions of bit value '1' in the codes. To design these codes, BIBD matrices are first used. Individual columns of BIBD matrices maintain unique bit patterns; when some column vectors have been overlapped and logical AND operations have been performed, the positions that have bit value '1' are unique. Original owners of the colluded content can be identified from these unique positions. Each BIBD

matrix is created with five parameters (v, b, r, k, λ) . The meanings of individual parameters are as follows.

v : Number of treatments

b : Number of blocks

r : Number of times each treatment is run, $k < v$

k : Number of treatments per block

λ : Number of blocks where a processing pair appears)

The five parameters satisfy the set under Expressions 9 through 12,

$$vr = bk \quad (9)$$

$$r(k-1) = \lambda(v-1) \quad (10)$$

$$b = \frac{v(v-1)\lambda}{k(k-1)} \quad (11)$$

$$r = \frac{\lambda(v-1)}{k-1} \quad (12)$$

and a BIBD code of size $v \times b$ is an Incidence Matrix M , of which the internal values are determined by Expression 13.

$$M = m_{ij} \quad (13)$$

$$m_{ij} = \begin{cases} 1 & \text{if } j_{th} \text{ blocks } \in i_{th} \text{ elements} \\ 0 & \text{otherwise,} \end{cases}$$

The row vectors of M become forensic marks given to b users and this can be used as a collusion-secure code. If the bit set of the column vectors of M is assigned to individual users, forensic marks that include '1' at unique positions during logical AND operations can be made from $(k-1)$ column vectors. For instance, the following matrix M shows a set of forensic marks using $(7, 4, 1)$ -BIBD.

$$M = \begin{pmatrix} 0001111 \\ 0110011 \\ 1010101 \\ 0111100 \\ 1100110 \\ 1011010 \\ 1101001 \end{pmatrix}$$

This code set can create and distribute 7-bit forensic marks to seven users and can extract conspirators for collusion attacks on any two-row vectors. If an averaging attack has been attempted on contents in which two forensic marks have been inserted, the forensic marks inserted from the colluded contents can be extracted and conspirators

can be identified by finding the points that have a value of ‘1’ at the same position as the extracted bit columns, using Expression 14. For instance, the first two columns in the above matrix were inserted into the contents bought by User 1 and User 2, respectively.

$$\begin{aligned} w_1 &= -s_1 - s_2 + s_3 - s_4 + s_5 + s_6 + s_7 \\ w_2 &= -s_1 + s_2 - s_3 + s_4 + s_5 - s_6 + s_7 \end{aligned} \quad (14)$$

If an averaging attack on these two codes is attempted using logical AND operations, the extracted code vector will become ‘-1000101’ by $-s_1 + 0 + 0 + 0 + s_5 + 0 + s_7$. Here, the fact that the conspirators are User 1 and User 2 can be identified from the fifth and seventh bit values of ‘1.’ That is, the fact that code vectors having ‘1’ as the value of the fifth and seventh bits is that only User 1 and User 2 can be seen. However, the method using BIBD matrices is not suitable for application in the current environment either because of the difficulty in code composition and the fact that, as the number of conspirators increases, the length of codes increases.

3 Secure code design using group

3.1 Group ACC design

Collusion-secure forensic marks can extract data from at least two conspirators within contents that were colluded and illegally distributed. However, these technologies involve problems such as the inability to apply code lengths, difficulty in securing large quantities of codes, or difficulty in composing codes. Therefore, these technologies cannot be applied to systems in Internet environments due to unspecified quantities of users.

To solve these problems, diverse algorithms have been proposed including some that expand existing BIBD matrices and some that use polynomial expressions. However, these proposals cannot be applied to systems that support unspecified quantities of users. Therefore, this paper proposes a technique that can compose group-based forensic mark sets using existing BIBD matrices, thereby creating efficiency.

A basic idea for extracting the data of at least one conspirator is creating unique codes for tracing them based on the location of the bit value ‘1’ in the codes. To design these codes, BIBD matrices from the paper written by Trappe [11] were used. Each row of the BIBD matrix maintains a unique bit pattern. When logical AND operations are performed by overlapping row vectors in a matrix, the location that has the bit value ‘1’ is unique. BIBD matrices are used to enable identification of the original owner of colluded contents. However, BIBD matrices have a drawback in that there are only seven collusion-secure codes that can be created using matrix M as conspirator-tracing codes. In addition, relatively complicated operations are required to compose collusion-secure codes.

In this paper, the concept of groups was introduced so that a collusion-secure code set W having seven elements can be made using the row vector of matrix M . This is created by using the existing (7,4,1) BIBD matrix to make groups named after the

Table 1 Set of grouping the ACC

		$W=\{x_1,x_2,\dots,x_n\}$	\parallel	$CW=\{y_1,y_2,\dots,y_n\}$
Group1	Member 1	x_1	\parallel	y_1
	\vdots	\vdots	\parallel	\vdots
	Member n	x_n	\parallel	y_n
Group2	Member 1	x_2	\parallel	y_1
	\vdots	\vdots	\parallel	\vdots
	Member n	x_2	\parallel	y_n
\vdots	\vdots	\vdots	\parallel	\vdots
Group n	Member 1	x_n	\parallel	y_1
	\vdots	\vdots	\parallel	\vdots
	Member n	x_n	\parallel	y_n

elements of set W , and assigning seven member codes per group for a total of 49 collusion-secure codes. In set W and set CW , copied from set W , if b is assumed to be an element of W and cb is assumed to be an element of CW , the combined set C can be obtained by substituting the element $p=(b, cb)$ of product set P . This is created by the equation $W \times CW$ and joined with the operation function $CAT(x, y)=x|y$. Therefore, sets can be created by grouping the ACC using the elements of W and CW , as shown in Table 1.

However, when making the above composition, a problem, as shown in Example 1, occurs.

Example 1: Group1- Member3($x_1 |y_3$) and Group2-Member4($x_2 |y_4$) made an average collusion attack using AND operations so that the following collusion code was created.

$$x_1 \cap x_2 |y_3 \cap y_4 = 0010111 \cap 0101101 |0111010 \cap 1001011 = 0010101 | 0001010$$

To review the resultant value, the 5th and 7th values of the groups become the 1st and 2nd elements with value ‘1’ of set W . Therefore, the colluded groups become the members of Groups 1 and 2. In addition, for individual members, the 4th and 6th elements of set W become the 3rd and 4th elements with value ‘1’ of the groups and the conspirators become Members 3 and 4.

However, in this example, conspirators cannot be identified because the groups to which they belong are unknown and this is against the permission of collusion, which is a requirement for forensic mark technologies.

To solve the problem as shown by the results of Example 1, it is assumed that an AND attack has been made on the element pair in a set, as shown by the following Expression 15.

$$\begin{aligned} A \parallel B, C \parallel D \\ A \parallel D, C \parallel B \end{aligned} \tag{15}$$

The resultant value from AND operations is shown by the following Expression 16.

$$\begin{aligned} A \parallel B \cap C \parallel D = A \cap C \parallel B \cap D \\ A \parallel D \cap C \parallel B = A \cap C \parallel D \cap B \end{aligned} \tag{16}$$

In this case, the commutative law of Boolean algebra creates the same resultant values. Therefore, a technique to prevent this is necessary. In this study, the problem was solved by adding verification codes. As principles for the verification codes, the theorem and law of Boolean algebra for logical operations in Table 2 were used.

Table 2 The theorem and law of Boolean algebra

Identity law	$A+0=A, A+1=1$	$A \cdot 1=A, A \cdot 0=0$
Idempotent law	$A+A=A$	$A \cdot A=A$
Complement Law	$A+\bar{A}=1$	$A \cdot \bar{A}=0$
Doublenegation Law	$\bar{\bar{A}}=A, \bar{\bar{\bar{A}}}=\bar{A}$	
Commutative Law	$A+B=B+A$	$A \cdot B=B \cdot A$
Associative Law	$A+(B+C)=(A+B)+C$	$A \cdot (B \cdot C)=(A \cdot B) \cdot C$
Distributed Law	$A \cdot (B+C)=AB+AC$	$A+B \cdot C=(A+B)(A+C)$
Absorption Law	$A+A \cdot B=A$	$A \cdot (A+B)=A$
De Morgan’s Law	$\overline{A+B}=\bar{A} \cdot \bar{B}$	$\overline{A \cdot B}=\bar{A} + \bar{B}$

Although there are associative and commutative laws, according to the definition of Boolean algebra, in the case of AND attacks and OR attacks, the associative and commutative laws can be prevented by using XOR operations. In the case of XOR attacks, verification bits can be added using AND operations or OR operations. For instance, according to the resultant values of the below Expressions 17 and 18, if AND operation verification bits are added to AND operation attacks, verification bits will not be formed because of the commutative law.

$$\begin{aligned}
 &A \parallel B \parallel A \cap B \\
 &\quad \text{AND} \\
 &C \parallel D \parallel C \cap D \\
 &= A \cap C \parallel B \cap D \parallel A \cap B \cap C \cap D
 \end{aligned} \tag{17}$$

$$\begin{aligned}
 &A \parallel D \parallel A \cap D \\
 &\quad \text{AND} \\
 &C \parallel B \parallel C \cap B \\
 &= A \cap C \parallel D \cap B \parallel A \cap D \cap C \cap B
 \end{aligned} \tag{18}$$

However, in the case of AND attacks, if XOR operations are used to create verification values, the values will be distinguished by the resultant values of Expressions 19 and 20.

$$\begin{aligned}
 &A \parallel B \parallel A \oplus B \\
 &\quad \text{AND} \\
 &C \parallel D \parallel C \oplus D \\
 &= A \cap C \parallel B \cap D \parallel A \oplus B \cap C \oplus D
 \end{aligned} \tag{19}$$

$$\begin{aligned}
 &A \parallel D \parallel A \oplus D \\
 &\quad \text{AND} \\
 &C \parallel B \parallel C \oplus B \\
 &= A \cap C \parallel D \cap B \parallel A \cap D \cap C \oplus B
 \end{aligned} \tag{20}$$

On the contrary, in the case of XOR attacks, AND or OR operations can be used for verification values because of Expressions 21 and 22.

$$\begin{aligned}
 & A\|B\|A\cap B \\
 & \oplus \\
 & C\|D\|C\cap D \\
 & = A\oplus C\|B\oplus D\|A\cap B\oplus C\cap D
 \end{aligned} \tag{21}$$

$$\begin{aligned}
 & A\|D\|A\cap D \\
 & \oplus \\
 & C\|B\|C\cap B \\
 & = A\oplus C\|D\oplus B\|A\cap D\oplus C\cap B
 \end{aligned} \tag{22}$$

Verification bits can be added to the individual codes as in Example 1 above, and as follows.

Example 2: If verification values are added to the user codes in Example 1, the code value of Group 1-Member3 will be as follows.

0010111 0111010 0101101(0010111 \oplus 0111010) 0101101(0010111 AND 0111010)

and the code value of Group 2-Member 4 will be as follows.

0101101 1001011 1100110(0101101 \oplus 1001011) 0001001(0101101 AND 1001011)

The collusion code created for individual members by AND operations will be as follows.

0000101 0001010 0100100 0001001

and by the collusion code, the first subset, {Group 1-Member 3, Group 2-Member 4} and the second subset {Group 1-Member 4, Group 2-Member 3} will be created. The verification values of individual subsets can be verified as follows.

The collusion code that can be obtained from the first subset will be as follows.

0000101 0001010 0100100 0001001

And the collusion code that can be obtained from the second subset will be as follows.

0000101 0001010 0010100 0000000

Because the collusion code created from the first subset is identical to the collusion code created by the members, the conspirator can be detected.

The results in Example 2 follow the theorem of Boolean algebra and were obtained using the conditions specifying that associative and commutative laws are valid in OR and AND operations, though invalid in XOR operations. This can be generalized as per Theorem 1.

Theorem 1: Group ACC set theorem

CAT(x,y)=x || y is a joint operation function

XOR(x,y)=x \oplus y is an XOR operation function

AND(x,y)=x.y is an AND operation function

W={x₁,x₂,...,x_n} is a collusion-secure code set

CW={y₁,y₂,...,y_n} is a copied collusion-secure code set

P=W \times CW={(a,b) | a \in W, b \in CW} is a product set of set W and set CW

Set CXOR and set CAND are obtained using the given functions and sets.

CXOR={ XOR(a,b) | a \in W, b \in W}

CAND={ AND(a,b) | a \in W, b \in W}

A group-based copy prevention code set group ACC can be obtained by substituting the individual obtained sets into the CAT function.

group ACC={ cat(s, q, r) | s \in p, q \in CXOR, r \in CAND }

A set with 49 collusion-secure codes can be created by obtaining seven collusion-secure codes and using the existing BIBD matrix according to Theorem 1. In addition, the concept of groups was introduced to the seven codes to obtain seven groups and collusion-secure code sets with seven members per group. Therefore, if collusion-secure code sets are used, collusion-secure codes that are larger by the square in number will be obtained. Using n pieces of collusion-secure code, $n*n$ pieces of code can be made if the proposed technique is used, while the quantity of collusion codes that can be obtained from a bit length of $4n$ using existing techniques is $4n$.

3.2 2-Detecting using group ACCs

Created group ACCs can be detected by reverse tracing code creation by theorem 1. The detecting algorithm is as follows.

Group ACC Detecting Algorithm:

Input: Collusion code, attack_type: AND attack or XOR attack

Output: Collusion user

```

detecting_CC(Collusion Code, attack_type) {
  find_group(group_bit) // finding group
  find_member(member_bit) // finding member
  if(AND_attack?){//attack_type is AND attack
    return find_user(CXOR_bit) // finding CXOR value using group and member
  }
  else if(XOR_attack?){//attack is XOR attack
    return find_user(CAND_bit) // finding CXOR value using group and member
  }
}

```

For instance, if it is assumed that User 1 and User 2 have been provided with Group 2-Member 3 codes and Group 3-Member 4 codes, respectively, and User 1 and User 2 made a collusion attack using AND operations, a collusion code will be produced as shown in Example 3.

Example 3:

User 1: 1011010 0101110 1110100 0001010

User 2: 0101110 1101001 1000111 0101110

Collusion code: 0001010 0101000 1000100 0001010

If conspirators are traced using the algorithm, Group 2 and Group 3 that have ‘1’ as the fourth and sixth values will be identified in the case of groups, and Member 3 and Member 4 that have ‘1’ as the second and fourth values will be identified in the case of members. However, two cases of collusion may occur as follows because Member 3 and Member 4 may be in Group 2 or Group 3.

Case 1:

Group2-Member3:1011010 0101110 1110100 0001010

Group3-Member4:0101110 1101001 1000111 0101110

Collusion code:0001010 0101000 1000100 0001010

Case 2:

Group2-Member4:1011010 1101001 0110011 1001000

Group3-Member3:0101110 0101110 0000000 0101110

Collusion code:0001010 0101000 0000000 0001000

If the collusion code in Example 3 is compared to the collusion codes in Case 1 and Case 2, Case 1 will be found to be identical to Case 3 because the third 7 bits of AND operation attacks are checked. Therefore, the fact that Group 2-Member 3 and Group 3-Member 4 colluded with other as shown in Case 1 can be identified, and this indicates that AND operation attacks are accurately traced.

In the case of XOR attacks, a collusion code as shown in Example 4 may be created.

Example 4:

User1: 1011010 0101110 1110100 0001010

User2: 0101110 1101001 1000111 0101110

Collusion code: 1110100 1000111 0110011 0100100

If conspirators are traced using the collusion code, Group 2 and Group 3 that have '0' as the fourth and sixth values will be identified in the case of groups, and Member 3 and Member 4 that have '0' as the second and fourth values will be identified in the case of members.

Case 3:

Group2-Member3:1011010 0101110 1110100 0001010

Group3-Member4:0101110 1101001 1000111 0101110

Collusion code:1110100 1000111 0110011 0100100

Case 4:

Group2-Member4:1011010 1101001 0110011 1001000

Group3-Member3:0101110 0101110 0000000 0101110

Collusion code:1110100 1000111 0110011 1100110

However, two cases of collusion may occur, as follows, because Member 3 and Member 4 may be in Group 2 or Group 3.

If the collusion code in Example 4 is compared to the collusion codes in Case 3 and Case 4, Case 3 will be found to be identical to Case 4 because the third 7 bits of XOR operation attacks are checked. Therefore, the fact that Group 2-Member 3 and Group 3-Member 4 colluded with other as shown in Case 3 can be identified and this indicates that XOR attacks are also accurately traced.

4 Performance evaluation

The algorithm proposed in this paper uses group ACCs, and it introduces the concept of grouping to create codes square with the number of existing codes. Using group ACC relieves the complexity of existing ACC code compositions using BIBD and improves the efficiency of codes. The efficiency of codes can be expressed with Expression 23 below.

$$\text{code rate} = \frac{b}{v} \quad (23)$$

Where v is the number of bits necessary to show the code of one user, b shows the number of the entire users. The efficiency of forensic marks means the number of users that can be accommodated by each code vector. The number and efficiency of forensic marks using BIBD and those of forensic marks using group ACC, which is the method presented in this paper, can be compared, as shown in Fig. 3. In Fig. 3, the x-axis shows the number of code vectors that represent all users and the y-axis

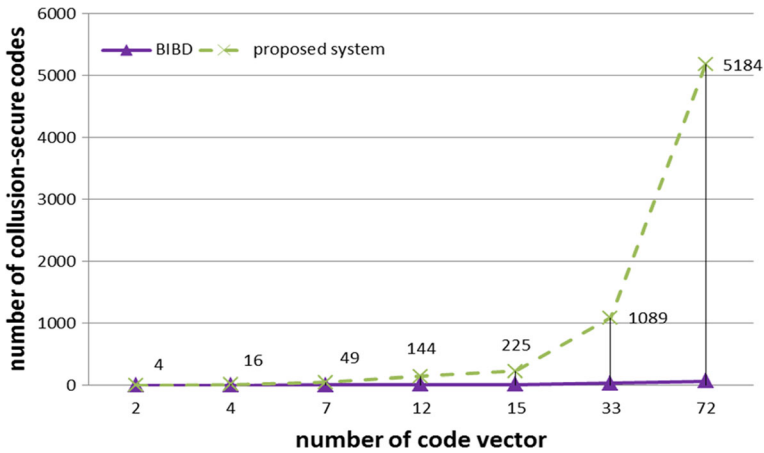


Fig. 3 Comparison of the numbers of collision-secure codes

shows the number of user codes that can be represented by the number of code vectors. When the number of code vectors is seven, where BIBD can create seven codes, the proposed system can create 49 codes.

When the number of code vectors is 33, where BIBD can create 33 codes, the proposed system can create 1,089 codes. As well, when the number of code vectors is 72, where BIBD can create 72 codes, the proposed system can create 5,184 codes. Therefore, although the numbers of codes that can be created are not very different from the number of code vectors, the differences become larger when the number of code vectors increases as the number of codes that can be created by the proposed system increases by square in number. Because the group ACC algorithm of the proposed system creates forensic marks using the existing BIBD, the number of collision-secure forensic marks becomes the square of b , which is the number

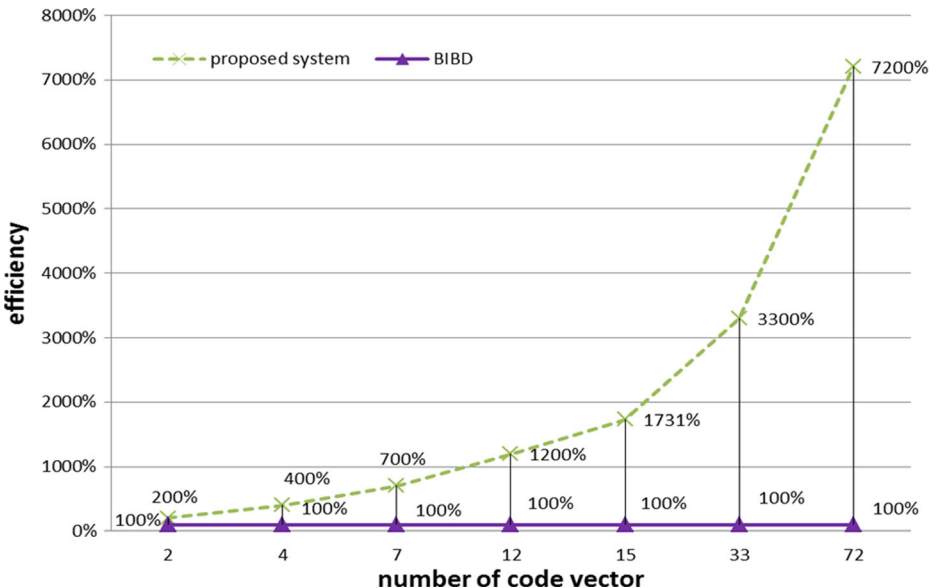


Fig. 4 The efficiency of collision-secure codes

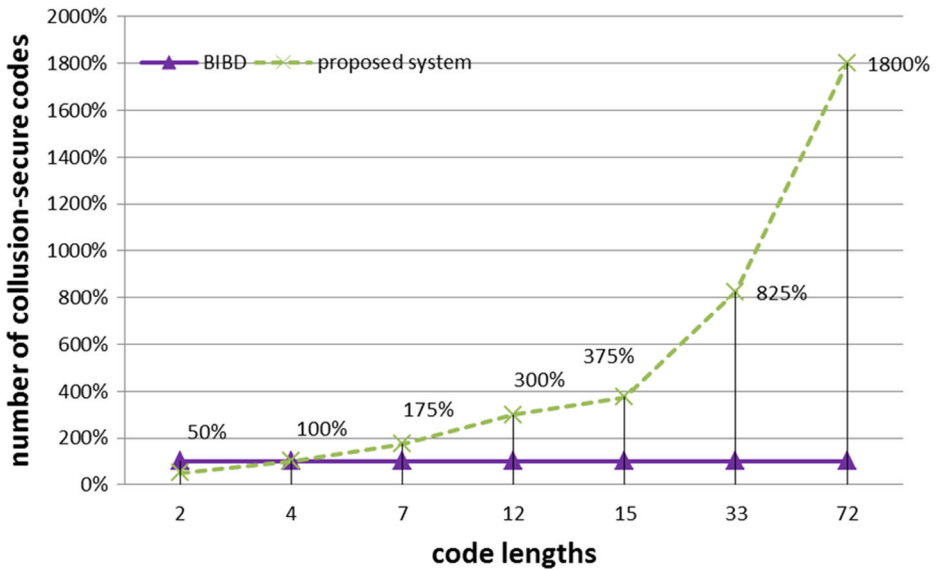


Fig. 5 Collusion-secure code efficiency in relation to code lengths

of collusion-secure forensic marks that can be created by the existing BIBD. The code efficiency of the algorithm proposed in this paper can be expressed by Expression 24.

$$\text{Group ACC code rate} = \frac{b^2}{v} \tag{24}$$

The code efficiency of the proposed system can be identified through Fig. 4. The x-axis shows the number of code vectors and the y-axis shows the efficiency. When the number of vectors is 72, where BIBD’s efficiency is 100 %, the proposed system’s efficiency is much higher, at 7,200 %.

However, the proposed algorithm has a weakness in that, although the number of elements v remains the same, the length of forensic marks increases to four times because group information and verification values for AND and XOR attacks are added. The magnitude of efficiency of collusion-secure codes is calculated by the number of code vectors in relation to code lengths, as shown in Fig. 5.

The fact is that when code lengths are below four, the efficiency of the existing BIBD is higher, and when code lengths are four or larger, the efficiency of the proposed algorithm is higher, as can be identified in Fig. 5. This indicates that, although code lengths increase slightly in the case of the proposed algorithm, the efficiency of collusion-secure code composition increases.

5 Conclusion

In this paper, a collusion-secure forensic mark design technique that introduces the concept of groups was proposed for the protection of digital contents. A group ACC algorithm that has more excellent expandability than existing collusion-secure forensic mark technologies expanded sets, of which the elements are the column vectors of given BIBD matrices, was proposed. This was in order to create group-based collusion-secure code sets, to separately create verification sets to solve the problem when two cases occur as group-based collusion-secure codes are introduced, and to obtain group ACC sets using the product and verification sets.

The element of a group ACC set becomes a forensic mark, and this recalled the difficulties in composing collusion-secure codes while solving the problem related to the number of collusion-secure codes in order to create codes that can be applied to diverse content protection environments. The fact that the efficiency of using group ACCs was higher than existing methods in terms of the number of codes and code composition could be identified.

Further studies are considered necessary on the construction of integrated forensic mark infrastructures that will enable the extraction of forensic marks by insertion algorithms.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Boneh D, Shaw J (1998) Collusion-secure fingerprinting for digital data. *IEEE Trans Int Theory* 44(5):1897–1905
2. Dittmann J, Behr A, Stabenau M, Schmitt P, Schwenk J, Ueberberg J (2000) Combining digital watermarks and collusion secure fingerprints for digital image. *J Electron Imaging* 9(4):456–467
3. Jiang X, Liu Q, Wu Q (2013) A new video watermarking algorithm based on shot segmentation and block classification. *Multimed Tools Appl* 62(3):545–560
4. Korea Copyright Commission (2010) Copyright technology of digital age
5. Korea Copyright Commission (2010) Content distribute path and response technology
6. Lian S, Chen X, Wang J (2012) Content distribution and copyright authentication based on combined indexing and watermarking. *Multimed Tools Appl* 57(1):49–66
7. Poon HT, Miri A, Zhao J (2009) An improved watermarking technique for multi-user, multi-right environments. *Multimed Tools Appl* 42(2):161–181
8. Stone H (1996) Analysis of attacks on image watermarks with randomized coefficients. *NEC Tech Rep*
9. Trappe W, Wu M, Wang J, Ray Liu KJ (2003) Anticollusion fingerprinting for multimedia. *IEEE Trans Signal Process* 51(4):1069–1087
10. Wahadaniah V, Guan YL, Chua HC (2002) A new collusion attack and its performance evaluation. *Proc IWDW* 88–103
11. Wu M, Trappe W, Wang ZJ, Ray Liu KJ (2004) Collusion-resistant fingerprinting for multimedia. *IEEE Signal Process Mag* 15–27
12. Yoshioka K, Shikata J, Matsumoto T (2003) Systematic treatment of collusion secure codes: security definitions and their relations. *Inf Secur Lect Notes Comput Sci* 2851:408–421



Youngmo Kim received his Ph.D. degree in Computer Engineering from Daejeon University, Daejeon, Korea in 2011. He is currently adjunct professor in Soongsil University and senior researcher Korea Copyright Commission. He is also working on several standardization activities and national project. His research interests are security, computer forensics, DRM(Digital Right Management), fingerprint.