Near-Optimal Detection for Binary Tardos Code by Estimating Collusion Strategy

Tatsuya Yasui, Minoru Kuribayashi, *Senior Member, IEEE,* Nobuo Funabiki, *Member, IEEE* and Isao Echizen, *Member, IEEE*

***Abstract*—A previously proposed optimal detector for bias-based fingerprinting codes such as Tardos and Nuida requires two kinds of important information: the number of colluders and the collusion strategy used to generate the pirated codeword. An estimator has now been derived for these two parameters. The bias in the pirated codeword is measured by observing the number of zeros and ones and compared with possible bias patterns calculated using information about the collusion strategy and number of colluders. Computer simulation demonstrated that the collusion strategy and number of colluders can be estimated with high probability and that the traceability of a detector using the proposed estimator is extremely close to being optimal.**

***Index Terms*—fingerprinting code, optimal detector, collusion strategy, number of colluders, estimator**

I. Introduction

In the field of collusion-secure codes [1]–[5], Tardos code [6] is known to produce bias-based fingerprinting in which each symbol of a colluder's codeword is determined by a certain biased probabilistic distribution. As the code length is theoretically of minimum order, the performance of a Tardos code has been intensively investigated to improve its traceability. In particular, Nuida et al. [7], [8] constructed an interesting variant using a discrete probabilistic (Gauss–Legendre) distribution to customize the bias-based fingerprinting code for a fixed number of possible colluders. For convenience, their fingerprinting code is referred to as the Nuida code in this paper.

To identify illegal users (colluders) from the pirated codeword, a tracing algorithm (detector) is used to find suspicious users by calculating the similarity among the colluder's codewords. Existing detectors can be classified into three types: catch-one, catch-many, and catch-all [9]. With the catch-one technique, the most suspicious user is the one with the maximum similarity score and is assumed to be guilty. The assumption here is that there is collusion among several illegal users, so a catch-many type detector is desirable because it can identify as many illegal users as possible. Although all colluders can be identified using a catch-all approach, the false-negative rate (i.e., no colluders are detected) is higher. Therefore, we focus here on catch-many detectors.

Tatsuya Yasui, Minoru Kuribayashi, and Nobuo Funabiki are with the Graduate School of Science and Technology, Okayama University, Okayama 700-8530, Japan. e-mail: yasui.tatsuya@s.okayama-u.ac.jp, kminoru@okayama-u.ac.jp

Isao Echizen is with the National Institute of Informatics, Tokyo, 101-8430, Japan.

A good tracing algorithm can catch as many colluders as possible with a constant and small false-positive rate. The tracing algorithm is essentially composed of two operations: scoring, which calculates similarity scores, and classification using a threshold. The colluders can choose an arbitrary collusion strategy, such as majority and minority voting, to generate a pirated codeword. As Tardos and its revised scoring functions [10] are independent of the collusion strategy, the functions cannot achieve high performance. Furon and Perez-Freire [11] proposed an optimal detector based on information theoretical analysis that calculates the highest score using information about the collusion strategy and the number of colluders. Because of the difficulty of estimating these parameters [11], several researcher groups [12]–[16] have investigated a defense strategy to minimize the performance gap from this optimal detector. For instance, scoring functions that adjust their weighting parameters on the basis of each symbol have been developed [15], [16]. These scoring functions require no information about the collusion strategy because they use only the symbol combination of the colluders' codewords and the pirated codeword.

We have developed an effective estimator for these parameters that uses the characteristics of the discretized probabilistic distribution of the Nuida code. A preliminary version of this paper appeared in the proceedings of APSIPA 2018 [17]. This estimator has two steps.

In the first step, the estimator observes the bias of the "1" symbols in the pirated codeword and then forms a feature vector in accordance with the characteristics of the bias-based fingerprinting code. Essentially, each symbol in the codeword is determined by each assigned bias probability as a secret sequence. Therefore, the bias of symbol "1" in each innocent user codeword is statistically stable and depends only on the bias probability. In contrast, the bias in the pirated codeword differs as it is affected by the collusion strategy and number of colluders. Because the number of candidates for the bias probability in the Nuida code is finite, the symbols in the pirated codeword can be classified into groups having the same bias probabilities. The expected probability of each of the symbols in a group becoming 1 after a collusion attack is then calculated. For each collusion strategy and number of colluders, almost all sets of expected probabilities are different. For convenience, such a set is defined as a Collusion Strategy Characteristic Vector (CSCV).

In the second step, the estimator identifies the CSCV closest to the feature vector, i.e., the one at a minimum distance from the CSCV, and estimates the collusion strategy and number

of colluders. Use of this technique enabled a detector to achieve estimation accuracy greater than 90% against seven well-known collusion strategies, with near-optimal traceability.

We also investigated a noisy case representing a realistic scenario [18], [19]. A codeword embedded in multimedia content using a watermarking technique was considered. If a pirated copy is produced by a coalition of illegal users, the codeword is further modified by signal processing operations such as lossy compression and filtering. This results in the addition of noise to the pirated codeword. As a result, the number of "1" and "0" symbols cannot be derived directly from the codeword. Thus, an additional estimator is required to adjust the parameters. The experimental results show that the traceability of the proposed method is still very near to optimal in the presence of noise.

The remainder of this paper is organized as follows. Section II reviews some basic information about fingerprinting codes and collusion attacks. Related studies are then discussed in Section III. Section IV introduces three proposed estimators. A noisy case is considered in Section V, and the experimental results are presented in Section VI. Finally, the key points are summarized in Section VII.

II. Fingerprinting Codes

This section reviews the process of constructing bias-based fingerprinting codes such as Tardos and Nuida and the design of tracing algorithms. It also introduces several collusion strategies that can be used to generate a pirated codeword.

A. Construction of Tardos Codes

A Tardos code is a binary bias-based fingerprinting code composed of N codewords with L symbols. Let $x_{j,i} \in \{0,1\}(1 \leq i \leq L)$ represent the j-th user's codeword, where $x_{j,i}$ is generated from an independent and identically distributed set of random numbers with probability p_i such that $\Pr[x_{j,i} = 1] = p_i$ and $\Pr[x_{j,i} = 0] = 1 - p_i$. This probability p_i needs to satisfy the following conditions, where the maximum number c_{max} of colluders should be determined during the construction of the codeword. We select p_i in accordance with continuous $f(p)$, where $f(p)$ is given by

$$f(p) = \frac{1}{2\sin^{-1}(1-2t)} \frac{1}{\sqrt{p(1-p)}}. \quad (1)$$

Both the codeword $\boldsymbol{x_j} = (x_{j,1}, x_{j,2}, \ldots, x_{j,L})$ and the sequence $\boldsymbol{P} = (p_1, p_2, \ldots, p_L)$ must be kept as secret parameters.

B. Construction of Nuida Codes

To improve the performance of the Tardos code, Nuida et al. presented a discrete version of the bias distribution that is customized for a given maximum number of colluders c_{max} [7], [8]. Let $L_k(x) = (\frac{d}{dx})^k (x^2-1)^k/(k!2^k)$ be the k-th Legendre polynomial, and set $\tilde{L}_k(x) = L_k(2x-1)$. We define $\mathcal{P}^{GL}_{2k-1} = \mathcal{P}^{GL}_{2k}$ to be the finite probability distribution whose values are the k zeros of $\tilde{L}_k$, with each value p selected with probability $\eta(p(1-p))^{-3/2}\tilde{L}'_k(p)^{-2}$, where η is a normalized constant that ensures the sum of the probabilities is equal to 1. Similar to the Tardos code, the codewords of the Nuida code are generated using the bias probability sequence $\boldsymbol{P}$. Because of the discrete values, the candidate values for $p_i \in \boldsymbol{P}$ are finite, and the number of candidates is $n_g = \lceil c_{max}/2 \rceil$. Each probability p_i can be classified into ξ groups. Numerical examples are presented in Table I, where P_ξ and Q_ξ, for $1 \leq \xi \leq n_g$, denote the values of the discretized probabilities and their emerging probabilities, respectively. For example, when $c_{max} = 8$ and length L of sequence $\boldsymbol{P}$ is 10000, the number of elements for which $p_i = P_2 = 0.33001$ is approximately $L \cdot Q_2 \approx 2517$ on average. As each symbol $x_{j,i}$ of the users' codewords is independently and identically selected under the constraint $\Pr[x_{j,i} = 1] = p_i$, the symbols of a codeword $\boldsymbol{x_j}$ can be separated into n_g groups on the basis of $p_i \in \boldsymbol{P}$.

TABLE I
Example of discrete Nuida code bias distribution.

c_{max}	P_ξ	Q_ξ	c_{max}	P_ξ	Q_ξ
1, 2	0.50000	1.00000	7, 8	0.06943	0.24833
3, 4	0.21132	0.50000		0.33001	0.25167
	0.78868	0.50000		0.66999	0.25167
5, 6	0.11270	0.33201		0.93057	0.24833
	0.50000	0.33598			
	0.88730	0.33201			

C. Collusion Attacks

Suppose that c colluders attempt to produce a pirated copy from their fingerprinting codes. Under the marking assumption [1], a pirated codeword $\boldsymbol{y} = (y_1, y_2, \ldots, y_L)$ is constructed using a collusion strategy. A group of colluders is denoted by $\boldsymbol{C} = \{j_1, j_2, \ldots, j_c\}$. The collusion attack is the process of taking sequences in $\boldsymbol{I_i} = \{x_{j_1,i}, x_{j_2,i}, \ldots, x_{j_c,i}\}$ as inputs and the pirated sequence $\boldsymbol{y}$ as an output. When a pirated codeword is produced from the colluders' codewords, the marking assumption [1] states that the colluders have $y_i \in \boldsymbol{I_i}$. They cannot change the bit in the position where all of the indexes in $\boldsymbol{I_i}$ are identical because their positions are undetectable.

Furon et al. defined a collusion attack as parameter vector $\boldsymbol{\theta^{str}_c} = (\theta^{str}_0, \ldots, \theta^{str}_c)$ with $\theta^{str}_\lambda = \Pr[y_i = 1 | \Phi = \lambda](0 \leq \lambda \leq c)$, where $\Phi \in \{0, \ldots, c\}$ denotes the number of "1" symbols in the colluders' copies for a given index [20]. Since some collusion strategies have a greater affect on traceability than others [20], the worst case attack (WCA), which minimizes the achievable rate of the code, can be defined from an information theoretical point of view. The marking assumption enforces $\theta^{str}_0 = 0$ and $\theta^{str}_c = 1$ in the collusion strategies. Typical examples for $c = 6$ are shown by the following parameters.

- Majority: $\boldsymbol{\theta^{maj}_6}$ = (0,0,0,0.5,1,1,1)
- Minority: $\boldsymbol{\theta^{min}_6}$ = (0,1,1,0.5,0,0,1)
- Coin-flip: $\boldsymbol{\theta^{coin}_6}$ = (0,0.5,0.5,0.5,0.5,0.5,1)
- All-0: $\boldsymbol{\theta^{all0}_6}$ = (0,0,0,0,0,0,1)
- All-1: $\boldsymbol{\theta^{all1}_6}$ = (0,1,1,1,1,1,1),

- Interleave: $\boldsymbol{\theta_6^{int}}$ = (0,$\frac{1}{6}$,$\frac{2}{6}$,$\frac{3}{6}$,$\frac{4}{6}$,$\frac{5}{6}$,1)
- WCA: $\boldsymbol{\theta_6^{WCA}}$ = (0, 0.5641, 0, 0.5, 1, 0.4359, 1)

For the above case, the candidates collusion strategies are denoted by $str = \{$maj, min, coin, all0, all1, int, WCA$\}$.

D. Tracing Algorithms

A tracing algorithm (a "detector") is composed of a scoring function and a classification function. We consider error rates ϵ_{FP} and ϵ_{FN}; tracing algorithm Tr outputs suspicious users, and $\boldsymbol{C}$ is the group of colluders.

- ϵ_{FP}: false positive

$$\epsilon_{FP} = \Pr[Tr(\boldsymbol{y}) \not\subset \boldsymbol{C} | Tr(\boldsymbol{y}) \neq \emptyset].$$

- ϵ_{FN}: false negative

$$\epsilon_{FN} = \Pr[Tr(\boldsymbol{y}) \cap \boldsymbol{C} = \emptyset].$$

Tardos proposed the following scoring function [6]:

$$S_j = \sum_{i=1}^{L} S_{j,i} = \sum_{i=1}^{L} y_i U_{j,i}, \tag{2}$$

where

$$U_{j,i} = \begin{cases} -\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i} = 1) \\ \sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i} = 0) \end{cases}. \tag{3}$$

For classification, the catch-one approach identifies the suspicious user with the maximum score as an illegal user if the score exceeds a threshold. The scoring function in Eq. (2) can be used for Nuida code. Unfortunately, the scoring function uses only half of the information about the pirated codeword because the value of score $S_{j,i}$ is zero when $y_i = 0$. To use all of the information, Škorić et al. [10] proposed using the symmetric version of the scoring function:

$$S_j^{sym} = \sum_{i=1}^{L} S_{j,i}^{sym} = \sum_{i=1}^{L} (2y_i - 1) U_{j,i}. \tag{4}$$

This function requires no information about the collusion strategy or the number c of colluders. To discriminate colluders from innocent users, an optimal scoring function should be designed using these parameters from an information theoretical point of view. Furon and Perez-Freire defined the optimal scoring function for a single detector as a log-likelihood ratio [11]:

$$S_j^{MAP} = \sum_{i=1}^{L} S_{j,i}^{MAP} = \sum_{i=1}^{L} \log\left(\frac{\Pr[y_i | x_{j,i}, \boldsymbol{\theta_c^{str}}]}{\Pr[y_i | \boldsymbol{\theta_c^{str}}]}\right), \tag{5}$$

where a single detector computes a score for each user while a joint detector computes a score for a subset of users. As this score represents the maximum a posteriori probability, the optimal scoring function is called the MAP detector. The denominator $\Pr[y_i | p_i, \boldsymbol{\theta_c^{str}}]$ can be calculated using

$$\begin{cases} \Pr[1 | \boldsymbol{\theta_c^{str}}] = \displaystyle\sum_{\lambda=0}^{c} \theta_\lambda^{str} \binom{c}{\lambda} p_i^\lambda (1-p_i)^{c-\lambda} \\ \Pr[0 | \boldsymbol{\theta_c^{str}}] = 1 - \Pr[1 | \boldsymbol{\theta_c^{str}}] \end{cases}. \tag{6}$$

Similarly, the numerator $\Pr[y_i | x_{j,i}, p_i, \boldsymbol{\theta_c^{str}}]$ can be calculated using

$$\begin{cases} \Pr[1|1, \boldsymbol{\theta_c^{str}}] = \displaystyle\sum_{\lambda=1}^{c} \theta_\lambda^{str} \binom{c-1}{\lambda-1} p_i^{\lambda-1} (1-p_i)^{c-\lambda} \\ \Pr[0|1, \boldsymbol{\theta_c^{str}}] = 1 - \Pr[1|1, \boldsymbol{\theta_c^{str}}] \\ \Pr[1|0, \boldsymbol{\theta_c^{str}}] = \displaystyle\sum_{\lambda=0}^{c-1} \theta_\lambda^{str} \binom{c-1}{\lambda} p_i^{\lambda} (1-p_i)^{c-\lambda-1} \\ \Pr[0|0, \boldsymbol{\theta_c^{str}}] = 1 - \Pr[1|0, \boldsymbol{\theta_c^{str}}] \end{cases}. \tag{7}$$

Moulin studied the theoretical aspect of a joint detector [21], and Meerwald and Furon proposed a practical implementation [13] that can be extended from a single detector. Therefore, we focus on a single detector here. Both theoretically and practically, the difficulty in designing such an optimal scoring function is how to estimate the collusion strategy str and the number of colluders c, namely $\boldsymbol{\theta_c^{str}}$, from a given codeword $\boldsymbol{y}$. Furon and Perez-Freire estimated these parameters using an expectation-maximization (EM) algorithm [11], but the accuracy of this approach is not high. To the best of our knowledge, there have been no other studies of the estimator.

E. Threshold

In a catch-many detector, suspicious users with scores exceeding a threshold Z are regarded as illegal users. Some methods approximate the distribution of a user's score S_j by using a Gaussian distribution [18] to calculate the threshold for satisfying a given false-positive probability. Any increase in the length of the users' codewords enhances the accuracy of the approximation. However, it has been reported [22] that such an approximation is not appropriate for calculating the threshold so that the false-positive rate is less than ϵ_{FP} because the tail of the Gaussian distribution is not accurate for short codewords. For accurate measurement in the tail part, Furon et al. [20] and Cérou et al. [23] proposed an efficient method for estimating the probability of rare events. By using this rare event simulator, we can estimate ϵ_{FP} for a given threshold Z, which means that we calculate the mapping $\epsilon_{FP} = F(Z)$.

III. Related Work

We first discuss the universal scoring function, which achieves better performance for an arbitrary collusion strategy than uninformed methods such as Škorić's symmetric scoring function [10]. Because of the difficulty of realizing the MAP detector, the scoring function has been adjusted so that a certain collusion strategy can achieve universality [12]–[16]. Bias in symbols "0" and "1" is observed, and the weights corresponding to the biases in Škorić's scoring function are used to calculate the score [24]. In this section, we review two scoring functions for our proposed estimator.

A. Bias Equalizer

In binary fingerprinting codes, the number of "0" and "1" symbols is balanced because of the symmetry of bias probability p_i. However, this balance is not always achieved in

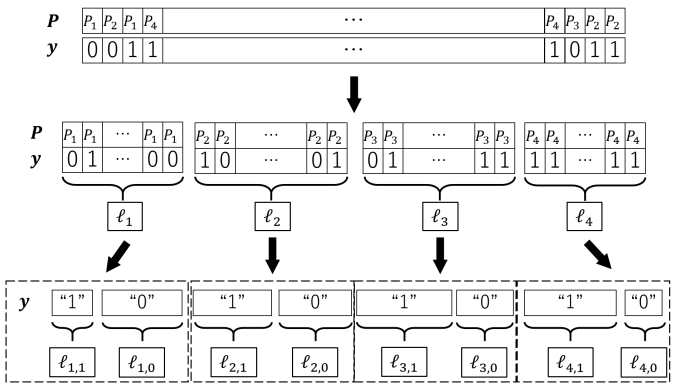


Fig. 1. Number of “0” and “1” symbols in pirated codeword.

a pirated codeword. Škorić's scoring function can be modified to compensate for the imbalance created by a collusion attack by equalizing the balance using weighting parameters, giving a “bias equalizer” [24]. Let $\mathcal{Y}_1$ and $\mathcal{Y}_0$ be the set of indices i satisfying $y_i = 1$ and $y_i = 0$, respectively. Then, the numbers of elements in $\mathcal{Y}_1$ and $\mathcal{Y}_0$ are denoted by L_1 and L_0, respectively, where $L_1 + L_0 = L$. Because of the symmetry of a bias distribution, it is expected that $L_1 = L_0$ unless the colluders do not know the actual values $x_{j,i}$ of their codewords. Therefore, in the case of $\boldsymbol{y}$ produced by “all-0” and “all-1,” L_1 is not always equal to L_0. As mentioned in Section II-B, each probability p_i can be classified into ξ groups. The number of elements in the ξ-th group is denoted by ℓ_ξ, where $\ell_\xi \geq 0$ and $\sum_{\xi=1}^{n_g} \ell_\xi = L$. Additionally, the number of “1” and “0” symbols are denoted by $\ell_{\xi,1}$ and $\ell_{\xi,0}$, respectively. Note that $\ell_{\xi,1} + \ell_{\xi,0} = \ell_\xi$. As an example, when $c_{max} = 8$, the classification of $n_g = 4$ groups is illustrated in Fig. 1. Using those parameters, the scoring function in the bias equalizer is as follows.

$$S^{Bias}_{j,i,\xi} = y_i \begin{cases} U^{00}_{j,i} = \dfrac{\ell_{\xi,1}}{\ell_\xi}\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i} = y_i = 0) \\ U^{01}_{j,i} = -\dfrac{\ell_{\xi,0}}{\ell_\xi}\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i} = 1, y_i = 0) \\ U^{10}_{j,i} = -\dfrac{\ell_{\xi,1}}{\ell_\xi}\sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i} = 0, y_i = 1) \\ U^{11}_{j,i} = \dfrac{\ell_{\xi,0}}{\ell_\xi}\sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i} = y_i = 1) \end{cases} \tag{8}$$

To adjust the above weighting parameters on the basis of the gap for each collusion strategy, the collusion strategy is classified into three types (all-0 or all-1 attack, minority or coin-flip attack, or other) for the bias equalizer [24]. First, the conditions in Eq. (9) were identified.

$$\begin{cases} \ell_{\xi,0} \approx \ell_\xi, & \text{if } p_i < 0.5 \text{ holds for all } \xi \\ \ell_{\xi,1} \approx \ell_\xi, & \text{if } p_i > 0.5 \text{ holds for all } \xi \end{cases} \tag{9}$$

All-0, all-1, and other strategies can be classified by introducing a threshold $T^\dagger$. For the classification of all-0 and all-1 attacks, the following two cases are checked.

$$\begin{cases} \dfrac{\ell_{\xi,0}}{\ell_\xi} > T^\dagger & (p_i < 0.5) \\ \dfrac{\ell_{\xi,1}}{\ell_\xi} > T^\dagger & (p_i > 0.5) \end{cases} \tag{10}$$

Note that $T^\dagger$ is close to 1 because of Eq. (9). In a previous study [24], threshold $T^\dagger$ was empirically determined to be 0.95. When the minority or coin-flip attack strategy is used, the following relations can be observed for the ξ-th group.

$$\begin{cases} \dfrac{\ell_{\xi,0}}{\ell_{\xi,1}} < \sqrt{\dfrac{1-p_i}{p_i}} & (p_i < 0.5) \\ \dfrac{\ell_{\xi,1}}{\ell_{\xi,0}} < \sqrt{\dfrac{p_i}{1-p_i}} & (p_i > 0.5) \end{cases} \tag{11}$$

Finally, a pirated codeword that has passed the above two steps is classified as being generated by one of the strategies used in majority, interleave, and worst case attacks. In this case, the collusion strategies are classified into one of three types, and an improved scoring function is used to revise the weights. Even though the bias equalizer improves the performance over that of uninformed scoring functions such as the symmetric decoder [10], the classification of collusion strategies is heuristic rather than theoretical. It is thus necessary to study a theoretical derivation for estimating collusion strategies.

B. Estimating the Number of Colluders

Information about the number of colluders is also required for scoring functions based on the MAP detector. For example, Meerwald et al. [13] assumed that the number of colluders is less than or equal to c_{max} and calculated the correlation scores for the number of colluders within $[1, c_{max}]$ for a scoring function based on the MAP detector. When the function is adjusted for WCA $\boldsymbol{\theta}^{\boldsymbol{WCA}}_{\boldsymbol{\lambda}}(1 \leq \lambda \leq c_{max})$, score $S^{WCA}_{j,i}$ is determined by the candidate with the maximum value.

$$S^{WCA}_j = \max_{1 \leq \lambda \leq c_{max}} \left(\sum_{i=1}^{L} \left(\log \frac{\Pr[y_i | x_{j,i}, \boldsymbol{\theta}^{\boldsymbol{WCA}}_{\boldsymbol{\lambda}}]}{\Pr[y_i | \boldsymbol{\theta}^{\boldsymbol{WCA}}_{\boldsymbol{\lambda}}]} \right) \right). \tag{12}$$

This kind of scoring function is called WCA defense because the score in Eq. (5) is oriented for a WCA attack. The method first calculates the c_{max} scores, from which the final score is produced. Thus, the number of colluders c is not directly estimated. Meerwald et al. also proposed a maximum likelihood estimator that guesses the collusion strategy $\boldsymbol{\theta}$ from a given pirated codeword.

IV. Proposed Estimator

This section describes how we estimate vector $\boldsymbol{\theta}^{\boldsymbol{str}}_{\boldsymbol{c}}$ for the optimal MAP detector. We exploit the bias in a pirated codeword to generate the estimate.

A. Collusion Strategy Characteristic Vector

When a pirated codeword is produced by a combination of codewords under the constraint of the marking assumption, the number of “0” and “1” symbols must have changed. We measure the number of changes on the basis of the discrete

TABLE II
COLLUSION STRATEGY CHARACTERISTIC VECTORS FOR $c = 6$.

$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{str}}$	ξ			
	$\gamma_{6,1}^{str}$	$\gamma_{6,2}^{str}$	$\gamma_{6,3}^{str}$	$\gamma_{6,4}^{str}$
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{maj}}$	0.00301	0.20498	0.79502	0.99699
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{min}}$	0.34762	0.70586	0.29414	0.65238
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{coin}}$	0.17531	0.45542	0.54458	0.82469
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{all0}}$	0.00000	0.00129	0.09045	0.64937
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{all1}}$	0.35063	0.90955	0.99871	1.00000
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{int}}$	0.06943	0.33001	0.66999	0.93057
$\boldsymbol{\Gamma}_{\boldsymbol{6}}^{\boldsymbol{WCA}}$	0.16699	0.34689	0.65311	0.83301

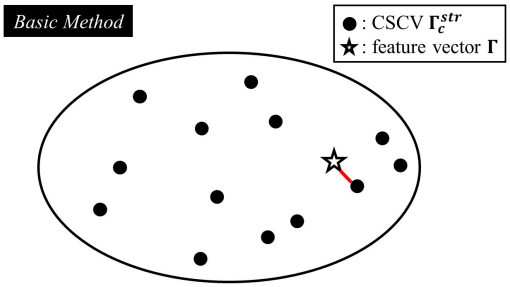


Fig. 2. Illustration of vector space $\mathbb{R}^{n_g}$ and estimation process in basic method.

bias probability. The emerging probability P_ξ, $(1 \le \xi \le n_g)$ is statistically equivalent to $\ell_{\xi,1}/\ell_\xi$ for each user's codeword. Hence, if we observe the number of symbols in a codeword, the following condition must be satisfied:

$$(P_1, \ldots, P_\xi, \ldots, P_{n_g}) \approx \left(\frac{\ell_{1,1}}{\ell_1}, \ldots, \frac{\ell_{\xi,1}}{\ell_\xi}, \ldots, \frac{\ell_{n_g,1}}{\ell_{n_g}} \right). \quad (13)$$

The right-hand term in Eq. (13) will be changed in a pirated codeword, and the number of changes in each element depends on the collusion strategy and the number of colluders. For convenience, the vector observed from a pirated codeword is denoted by

$$\boldsymbol{\Gamma} = (\gamma_1, \ldots, \gamma_\xi, \ldots, \gamma_{n_g}), \quad (14)$$

where

$$\gamma_\xi = \frac{\ell_{\xi,1}}{\ell_\xi}. \quad (15)$$

The expectation of the elements in $\boldsymbol{\Gamma}$ can be calculated from $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$ and c as

$$\gamma_{c,\xi}^{str} = \sum_{\lambda=0}^{c} \binom{c}{\lambda} P_\xi^\lambda (1 - P_\xi)^{c-\lambda} \theta_\lambda^{str}. \quad (16)$$

The vector $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}} = (\gamma_{c,1}^{str}, \ldots, \gamma_{c,\xi}^{str}, \ldots, \gamma_{c,n_g}^{str})$ is called the CSCV. Under the marking assumption, Eq. (16) enables us to express $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ for every general collusion strategy we can conceive. Several example collusion strategies are listed in Table II, where $c_{max} = 8$ for the Nuida code and the actual number of colluders c is 6. We store the CSCVs $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}} (\tilde{c}_{min} \le c \le \tilde{c}_{max})$ with the general collusion strategy into a database, where $\tilde{c}_{min}$ and $\tilde{c}_{max}$ are the assumed minimum and maximum number of colluders, respectively. Apart from these thresholds, we can measure the distance from feature vector $\boldsymbol{\Gamma}$ to each CSCV and find the closest $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ for each possible collusion strategy.

B. Vector Space

We define a vector space $\mathbb{Z}_2^L$ for a codeword represented by a binary vector of length L. The calculation of CSCV $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ from a given pirated codeword can be regarded as the mapping from vector space $\mathbb{Z}_2^L$ to a rational vector space with n_g dimension $\mathbb{R}^{n_g}$. The CSCV's $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ can be derived from the following reduced-dimension map f in accordance with $L \gg n_g$.

$$f : \mathbb{Z}_2^L \mapsto \mathbb{R}^{n_g} \quad (17)$$

The symmetric decoder computes the correlation score between the pirated codeword and the users' codewords. As mentioned in Section II-D, the realization of the MAP detector depends on how we estimate the collusion strategy and the number of colluders. Therefore, the map Eq. (17) implies that the detection of colluders can be performed in a lower dimension.

C. Basic Method

Let $D^{str,c}$ be the distance between the observed vector $\boldsymbol{\Gamma}$ and all CSCVs $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$. Note that $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ can be calculated using Eq. (16) in advance and stored in a database. The basic method finds the closest vector $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ that minimizes distance D_c^{str}.

Well-known metrics for distance are the Total Variation distance D_1, the Euclidean distance D_2, and the Hellinger distance D_{Hel}:

$$D_1^{str,c} = \sum_\xi |\gamma_{c,\xi}^{str} - \gamma_\xi|, \quad (18)$$

$$D_2^{str,c} = \sqrt{\sum_\xi \left(\gamma_{c,\xi}^{str} - \gamma_\xi \right)^2}. \quad (19)$$

$$D_{Hel}^{str,c} = \sqrt{\sum_\xi \left(\sqrt{\gamma_{c,\xi}^{str}} - \sqrt{\gamma_\xi} \right)^2}. \quad (20)$$

In estimating the collusion strategy and number of colluders, we calculate $D^{str,c}$ for all CSCVs and find the combination that minimizes $D^{str,c}$ in vector space $\mathbb{R}^{n_g}$:

$$(\tilde{str}, \tilde{c}) = \arg\min_{str,c} D^{str,c}. \quad (21)$$

Fig. 2 shows vector space $\mathbb{R}^{n_g}$ and illustrates the estimation process in the basic method. The user's score S_j^{basic} is then calculated using

$$S_j^{basic} = \sum_{i=1}^{L} S_{i,j}^{basic} = \sum_{i=1}^{L} \log \left(\frac{\Pr[y_i | x_{j,i}, \boldsymbol{\theta}_{\tilde{\boldsymbol{c}}}^{\boldsymbol{s\tilde{t}r}}]}{\Pr[y_i | \boldsymbol{\theta}_{\tilde{\boldsymbol{c}}}^{\boldsymbol{s\tilde{t}r}}]} \right). \quad (22)$$

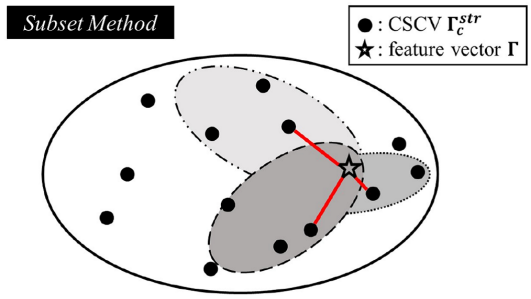


Fig. 3. Illustration of estimation process in subset method.

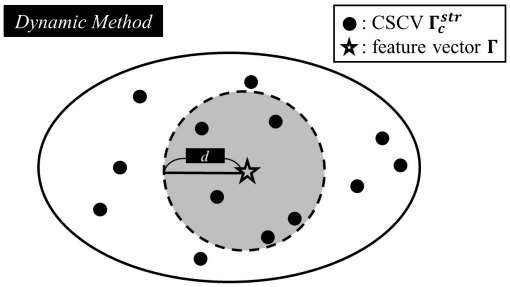


Fig. 4. Illustration of estimation process in dynamic method.

D. Subset Method

The proposed estimator searches for possible collusion attacks using the CSCVs stored in a database using a somewhat exhaustive search. Instead of a fully exhaustive search, the subset method calculates a set of user scores for some candidate number of colluders and outputs the number that maximizes the score.

The vector space $\mathbb{R}^{n_g}$ of all CSCVs is first separate into c subsets. A candidate CSCV can then be estimated in the subset. Fig. 3 illustrates the estimation process. An improvement in the estimation accuracy can be expected because the number of candidates $\tilde{c}_{max} - \tilde{c}_{min}$ in the subsets is reduced. Finally, the candidate user scores $\tilde{S}_{j,i}^{c}$ are calculated for estimated collusion strategy $\tilde{str}$ and for the number of colluders c corresponding to its subset. The maximum score is then determined as the user's score S_j^{sub}. This process is summarized as follows:

1) Initialize $c = \tilde{c}_{min}$.
2) Perform the following operations until $c = \tilde{c}_{max}$.
   2-1) Estimate strategy $\tilde{str}$ using Eq. (21) by fixing $\tilde{c} = c$.
   2-2) Increment $c = c + 1$.
   2-3) Using estimated vector $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$, calculate $S_{j,i}^{c}$ for $1 \leq i \leq L$ as

$$S_{j,i}^{c} = \log\left(\frac{\Pr[y_i|x_{j,i}, \boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}]}{\Pr[y_i|\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}]}\right). \tag{23}$$

3) Calculate total score S_j^{sub} by summarizing maximum scores $S_{j,i}^{c}$ for $1 \leq i \leq L$.

$$S_j^{sub} = \max_{\tilde{c}_{min} \leq \lambda \leq \tilde{c}_{max}} \left(\sum_{i=1}^{L} S_{j,i}^{\lambda}\right) \tag{24}$$

As $\tilde{c}_{max} - \tilde{c}_{min}$ increases, the computational cost increases linearly because step 2 is repeated $\tilde{c}_{max} - \tilde{c}_{min}$ times. For example, if $\tilde{c}_{min} = 2$ and $\tilde{c}_{max} = 10$, the computational cost of the subset method is nine times greater than that of the basic method. However, the subset method achieves higher estimation accuracy of the collusion strategy in each subset because the number of candidates for estimation in a subset is fewer than in the basic method's set.

E. Dynamic Method

A preliminary experiment showed that the number of colluders detected by the subset method is greater than in the basic method though the computational cost is proportional to the number of candidate vectors $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$. In short, there is a trade-off between computational cost and traceability. Hence, we consider a new method that changes the number of $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$ dynamically while maintaining detection accuracy. This is called the dynamic method.

In vector space $\mathbb{R}^{n_g}$, we introduce an $(n_g - 1)$-hypersphere $\Omega^{n_g - 1} = \{\boldsymbol{z} \in \mathbb{R}^{n_g} : \|\boldsymbol{z}\| = d\}$, where the radius d is a given positive number. For $n_g = 2$ and $n_g = 3$, the 1-hypersphere Ω^1 and 2-hypersphere Ω^2 are called a circle and a sphere, respectively. As mentioned in Section IV-B, any CSCV can be expressed by points in vector space $\mathbb{R}^{n_g}$, and the vector $\mathbf{\Gamma}$ derived from a pirated codeword can be placed at a certain point in $\mathbb{R}^{n_g}$. When we consider the $(n_g - 1)$-hypersphere $\Omega^{n_g - 1}$ centered at a point in $\mathbf{\Gamma}$, one of the CSCV candidates around $\mathbf{\Gamma}$ should have a high probability of being correct. The CSCV candidates for vector space $\mathbb{R}^{n_g}$ are illustrated in Fig. 4. The score in the dynamic method is calculated as follows:

1) Calculate distances $D^{str,c}$ for all possible CSCVs.
2) Form a set $\mathcal{D}$ of pairs (str, c) for which $D^{str,c}$ is less than d.
3) Perform the following operations if $\mathcal{D} \neq \{\text{null}\}$.
   3-1) Calculate scores $S_{j,i}^{str,c}$ with $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$ for all pairs $(str, c) \in \mathcal{D}$.

$$S_{j,i}^{str,c} = \log\left(\frac{\Pr[y_i|x_{j,i}, \boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}]}{\Pr[y_i|\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}]}\right) \tag{25}$$

   3-2) Calculate total score $S_j^{dynamic}$ by summing maximum scores $S_{j,i}^{str,c}$ for $1 \leq i \leq L$.

$$S_j^{dynamic} = \max_{(str,c) \in \mathcal{D}} \left(\sum_{i=1}^{L} S_{j,i}^{str,c}\right) \tag{26}$$

4) If $\mathcal{D} = \{\text{null}\}$, $S_j^{dynamic} = S_j^{basic}$.

V. Practical Situations

In practical situations, the pirated codeword may be distorted by noise. Noise can be modeled as additive white Gaussian noise (AWGN) [19]. This section shows how we estimate the collusion strategy and the number of colluders using CSCVs from a pirated codeword distorted by AWGN.

First, an EM algorithm is applied to all symbols of the pirated codeword for estimating noise variance σ_e^2. Then, γ_ξ is estimated from the symbols in the ξ-th group by using the estimated variance.

A. Estimating Number of Symbols

As mentioned in Section III-A, the number of "0" and "1" symbols has a different bias for each group. In the noiseless case, we can observe the bias by directly counting $\ell_{\xi,0}$ and $\ell_{\xi,1}$ and derive the feature vector $\mathbf{\Gamma}$ as given by Eq. (14).

In the noisy case, two estimation processes are required to derive $\ell_{\xi,0}$ and $\ell_{\xi,1}$ from a distorted codeword:

$$\hat{y}_i = y_i + e_i, \tag{27}$$

where e_i is AWGN with variance σ_e^2. The probability density function $pdf(\hat{y}_i)$ is given by the following equation as a Gaussian mixture model (GMM):

$$pdf(\hat{y}_i) = \pi_0 \mathcal{N}\big(\hat{y}_i; 0, \sigma_e^2\big) + \pi_1 \mathcal{N}\big(\hat{y}_i; 1, \sigma_e^2\big), \tag{28}$$

where $\sum_{k=0}^{1} \pi_k = 1$ and π_k represent the weights of each distribution, and

$$\mathcal{N}\big(\hat{y}_i; \mu, \sigma_e^2\big) = \frac{1}{\sqrt{2\pi\sigma_e^2}} \exp\Big(- \frac{(\hat{y}_i - \mu)^2}{2\sigma_e^2}\Big). \tag{29}$$

First, we estimate σ_e^2, π_0, and π_1 from all symbols $\hat{y}_i (1 \le i \le L)$ in the distorted codeword using the EM algorithm. The estimated variance is denoted by $\tilde{\sigma}_e^2$. Then, for the ξ-th group, γ_ξ is estimated from $\ell_\xi = \ell_{\xi,0} + \ell_{\xi,1}$ symbols using the EM algorithm. As the symbols $\hat{y}_i$ are distorted by noise, the probability density function in the ξ-th group is

$$\begin{aligned} pdf(\hat{y}_i)_\xi &= \frac{\ell_{\xi,0}}{\ell_\xi} \mathcal{N}\big(\hat{y}_i; 0, \tilde{\sigma}_e^2\big) + \frac{\ell_{\xi,1}}{\ell_\xi} \mathcal{N}\big(\hat{y}_i; 1, \tilde{\sigma}_e^2\big), \\ &= (1 - \gamma_\xi) \mathcal{N}\big(\hat{y}_i; 0, \tilde{\sigma}_e^2\big) + \gamma_\xi \mathcal{N}\big(\hat{y}_i; 1, \tilde{\sigma}_e^2\big) \end{aligned} \tag{30}$$

As the variance is estimated in the first process, the EM algorithm estimates γ_ξ in the second process. As a consequence, feature vector $\hat{\mathbf{\Gamma}}$ is calculated from the distorted codeword $\hat{\boldsymbol{y}}$.

B. Optimal Detection in a Noisy Environment

As feature vector $\hat{\mathbf{\Gamma}}$ is distorted by noise, distance $D^{str,c}$ changes accordingly. We assume that the additive noise follows a white Gaussian distribution in the CSCV vector space. Therefore, the feature vector $\hat{\mathbf{\Gamma}}$ can be estimated using CSCVs in a noiseless environment.

However, the distortions in the pirated codeword change vector $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$. Hence, the MAP detector must adjust vector $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$ in accordance with the noise variance estimated from the pirated codeword $\hat{\boldsymbol{y}}$. As discussed by Meerwald and Furon [13], the adjusted parameters $\hat{\theta}_\lambda (0 \le \lambda \le c)$ are given by

$$\hat{\theta}_{\lambda(\hat{y}_i)} = (1 - \theta_\lambda) \mathcal{N}\big(\hat{y}_i; 0, \tilde{\sigma}_e^2\big) + \theta_\lambda \mathcal{N}\big(\hat{y}_i; 1, \tilde{\sigma}_e^2\big). \tag{31}$$

After the above adjustment of collusion strategy $\hat{\boldsymbol{\theta}}_{\boldsymbol{c}}^{\boldsymbol{str}}$, we can execute the methods described in Section IV.

VI. Experimental Results

We ran simulation experiments to compare the performance of the three proposed methods. The experimental setup was as follows. The number of users in a system was $N = 10^6$. The Nuida code was designed using $c_{max} = 8$, and the number of candidate values n_g for p_i was 4. The vector space of codeword $\mathbb{Z}_2^L$ as mapped to $\mathbb{R}^4$ to calculate the CSCVs. The false-positive probability was set to $\epsilon = 10^{-10}$ and $\epsilon_{FP} = (1 - \epsilon)^N \approx N\epsilon = 10^{-4}$ using a rare event simulator [20], [23]. The candidate collusion strategies were $str = \{$maj, min, coin, all0, all1, int, WCA$\}$, and the number of colluders ranged from $\tilde{c}_{min} = 2$ to $\tilde{c}_{max} = 10$. There were 63 CSCVs $(= 7 \times 9)$. The pirated codewords were produced by a collusion attack on 10^2 randomly selected combinations of c colluders.

A. Estimation Accuracy of Collusion Strategy

Assuming that the number of colluders c is known in advance, the accuracy of the estimator in the subset method can be measured. The collusion strategies are exactly the same or almost the same in some cases. When they are, the θ values are coincident. Thus, it is not necessary to distinguish such a strategy. Nevertheless, Table III lists the accuracy with which the collusion strategy was estimated for distances $D_1^{str,c}$ and $D_2^{str,c}$ for $2 \le c \le 8$. For $c = 2$, the CSCVs calculated using maj, min, coin, int, and WCA were exactly the same, and the strategies were estimated without error. The greater the number of colluders and the longer the code, the greater the accuracy. Additionally, the estimation for each code length was highly accurate. Comparing the results for $L = 1024$ and $L = 4096$, we see that these code lengths are sufficient for estimating the collusion strategy. For $c = 4$, $\boldsymbol{\theta}^{\boldsymbol{coin}}$ and $\boldsymbol{\theta}^{\boldsymbol{WCA}}$ were very close but not exactly same. The accuracy was lower due to the similarity between $\mathbf{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{coin}}$ and $\mathbf{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{WCA}}$. Table IV compares $\mathbf{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{maj}}$, $\mathbf{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{coin}}$, and $\mathbf{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{WCA}}$. Our preliminary experiment showed that the effect of misestimation for coin-flip and WCA was much lower than for the other strategies. As observed from Table III, there was no remarkable difference among three metrics for distance. Therefore, we used $D_1^{str,c}$ in the following experiments because of its calculation simplicity.

B. Determination of Radius for Dynamic Method

To use the dynamic method, it is necessary to determine radius d. As shown in Fig. 5, $d = 0.102$ gives the maximum traceability. The main purpose of the dynamic method is to reduce the computational cost while maintaining performance. To compare the computational cost, we measured the number n_c^{str} of CSCVs within radius $d = 0.102$ for each $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$ and calculated its average

$$\bar{n}^{str} = \frac{1}{\tilde{c}_{max} - \tilde{c}_{min}} \sum_{c=\tilde{c}_{min}}^{\tilde{c}_{max}} n_c^{str} \tag{32}$$

for each strategy using the dynamic method. As the computational cost for the subset method is rational to the number

TABLE III
ACCURACY OF ESTIMATOR WHEN c IS KNOWN.

(a) $L = 1024$

$\boldsymbol{\theta_c^{str}}$		number c of colluders						
		2	3	4	5	6	7	8
maj	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	99.9	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
min	$D_1^{str,c}$	-	99.0	100	100	100	100	100
	$D_2^{str,c}$	-	99.2	100	100	100	100	100
	$D_{Hel}^{str,c}$	-	99.3	100	100	100	100	100
coin	$D_1^{str,c}$	-	70.2	58.3	94.9	98.9	100	100
	$D_2^{str,c}$	-	71.1	60.4	94.9	98.9	100	100
	$D_{Hel}^{str,c}$	-	68.3	59.8	94.9	98.8	100	100
int	$D_1^{str,c}$	-	89.7	99.0	100	100	100	100
	$D_2^{str,c}$	-	87.6	98.5	99.9	100	100	100
	$D_{Hel}^{str,c}$	-	88.1	98.2	99.6	99.9	100	100
all0	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
all1	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
WCA	$D_1^{str,c}$	-	77.7	59.3	92.8	98.6	100	100
	$D_2^{str,c}$	-	78.8	60.6	93.7	99.0	100	100
	$D_{Hel}^{str,c}$	-	77.0	58.6	92.3	98.5	99.9	100
average	$D_1^{str,c}$	100	90.9	88.1	98.2	99.6	100	100
	$D_2^{str,c}$	100	90.9	88.5	98.4	99.7	100	100
	$D_{Hel}^{str,c}$	100	90.4	88.1	98.1	99.6	99.9	100

(b) $L = 2048$

$\boldsymbol{\theta_c^{str}}$		number c of colluders						
		2	3	4	5	6	7	8
maj	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
min	$D_1^{str,c}$	-	99.8	100	100	100	100	100
	$D_2^{str,c}$	-	99.8	100	100	100	100	100
	$D_{Hel}^{str,c}$	-	99.8	100	100	100	100	100
coin	$D_1^{str,c}$	-	86.7	65.1	98.9	99.9	100	100
	$D_2^{str,c}$	-	85.2	67.5	98.9	99.9	100	100
	$D_{Hel}^{str,c}$	-	83.5	63.7	99.1	100	100	100
int	$D_1^{str,c}$	-	97.2	100	100	100	100	100
	$D_2^{str,c}$	-	96.8	99.8	100	100	100	100
	$D_{Hel}^{str,c}$	-	96.7	99.9	100	100	100	100
all0	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
all1	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
WCA	$D_1^{str,c}$	-	87.6	65.9	99.2	100	100	100
	$D_2^{str,c}$	-	88.3	67.7	99.3	100	100	100
	$D_{Hel}^{str,c}$	-	87.9	66.3	98.8	100	100	100
average	$D_1^{str,c}$	100	95.9	90.1	99.7	99.9	100	100
	$D_2^{str,c}$	100	95.7	90.7	99.7	99.9	100	100
	$D_1^{str,c}$	100	95.4	90.0	99.7	100	100	100

(c) $L = 4096$

$\boldsymbol{\theta_c^{str}}$		number c of colluders						
		2	3	4	5	6	7	8
maj	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
min	$D_1^{str,c}$	-	100	100	100	100	100	100
	$D_2^{str,c}$	-	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	-	100	100	100	100	100	100
coin	$D_1^{str,c}$	-	97.3	71.6	100	100	100	100
	$D_2^{str,c}$	-	96.9	73.6	100	100	100	100
	$D_{Hel}^{str,c}$	-	94.8	70.4	99.9	100	100	100
int	$D_1^{str,c}$	-	99.7	100	100	100	100	100
	$D_2^{str,c}$	-	99.7	100	100	100	100	100
	$D_{Hel}^{str,c}$	-	99.7	100	100	100	100	100
all0	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
all1	$D_1^{str,c}$	100	100	100	100	100	100	100
	$D_2^{str,c}$	100	100	100	100	100	100	100
	$D_{Hel}^{str,c}$	100	100	100	100	100	100	100
WCA	$D_1^{str,c}$	-	96.2	73.0	99.9	100	100	100
	$D_2^{str,c}$	-	96.2	72.8	99.9	100	100	100
	$D_{Hel}^{str,c}$	-	95.1	71.6	100	100	100	100
average	$D_1^{str,c}$	100	99.0	92.1	99.9	100	100	100
	$D_2^{str,c}$	100	99.0	92.3	99.9	100	100	100
	$D_1^{str,c}$	100	98.5	91.7	99.9	100	100	100

TABLE IV
COMPARISON OF VALUES CALCULATED USING ESTIMATOR FOR $c = 4$ (MAJORITY, COIN-FLIP, WCA).

str	$\gamma_{4,1}$	$\gamma_{4,2}$	$\gamma_{4,3}$	$\gamma_{4,4}$
majority	0.01379	0.25484	0.74516	0.98621
coin-flip	0.12507	0.40518	0.59482	0.87493
WCA	0.11801	0.39565	0.60435	0.88199

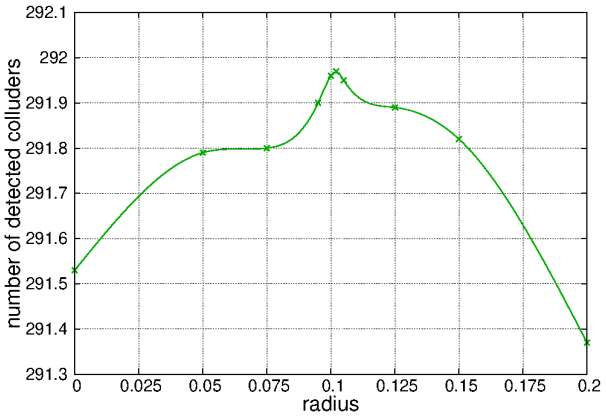


Fig. 5. Number of detected colluders versus radius d.

$(\tilde{c}_{max} - \tilde{c}_{min})$, as explained in Section IV-D, the cost ratio R^{str} was calculated for the comparison.

$$R^{str} = \frac{\bar{n}^{str}}{\tilde{c}_{max} - \tilde{c}_{min}} \tag{33}$$

Table V presents the number n_c^{str} of CSCVs for $2 \leq c \leq 10$, its average $\bar{n}^{str}$, and the cost ratio R^{str}. Clearly, the dynamic method reduces the computational cost with little sacrifice in performance.

C. Traceability

Table VI presents the sum of detected colluders for $2 \leq c \leq 10$, where the maximum is 54 = $\sum_{c=2}^{10} c$. With the MAP detector, the collusion strategy and number of colluders are known, so the number of detected colluders for MAP is the theoretical upper limit. Our estimator finds the closest CSCV among finite candidates, which are the well-known seven strategies and number of colluders. Hence, we also should show the performance of the detector when colluders attempt to attack using an out-of-list strategy so that misestimation occurred in our estimator. As it is difficult to check all possibilities, we checked the impact of misestimation for three collusion strategies:

- mix 1: $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{mix1}} = (\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{int}} + \boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{WCA}})/2$
- mix 2: $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{mix2}} = (\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{int}} + \boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{maj}})/2$
- mix 3: $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{mix3}} = (\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{int}} + \boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{coin}})/2$

Table VI (c) shows the traceability with these strategies. The results indicate that the traceability was very close to that of the optimal detector informing the actual strategy. From the results, we can say that, if the feature vector of the pirated codeword is close to one of the CSCVs of the seven strategies, the traceability is still close to that of the optimal detector. Since our final goal is to catch as many colluders as possible, a mismatch in estimating the collusion strategy is not a problem if the traceability is very close to that of the optimal detector.

As shown in Table VI, the results with the proposed methods exceeded the number of detectors in the optimal single detector for some cases. This is because of the probabilistic algorithm in the rare event simulator [20], [23] used to calculate the threshold. If the number of trials were increased, this would not occur. The table also shows that the traceability of the basic, subset, and dynamic methods were very close to that of the optimal MAP detector for all collusion strategies. When $L = 2048$, the subset method outperformed the other methods.

D. Noisy Case

The total number of detected colluders for the noisy environment case are listed in Table VII. The signal-to-noise ratio (SNR) ranged from 0 to 10 [dB], and the number of colluders was set to 6. The values were at most 66 (= 6×11). For the MAP detector, the collusion strategy, number of colluders, and AWGN variance were considered known. To further evaluate the accuracy of the proposed methods, we used the variance estimated by the EM algorithm in the MAP detector. Unlike the noiseless case, the total traceability of the dynamic method was better than that of the subset method in the presence of noise. The results of these experiments are illustrated in Fig. 6. These results clearly show that an optimal detector can be achieved by using the proposed estimators for the collusion attack parameters in the presence of noise.

VII. CONCLUSION

The three methods proposed for estimating collusion attack parameters for the optimal MAP detector use the imbalance between "0" and "1" symbols in the pirated codeword. In accordance with the discrete bias probabilities in the Nuida code, the imbalances are calculated for possible collusion strategies and numbers of colluders as a Collusion Strategy Characteristic Vector. In the estimation stage, the distances between the imbalances observed in the pirated codeword and the CSCVs stored in a database are examined, and the closest one is identified as a colluder. Computer simulation demonstrated that the overall performance of the proposed methods was superior to that of conventional techniques and was very close to optimal MAP performance.

ACKNOWLEDGMENTS

The authors thank Dr. Teddy Furon, Rennes INRIA, France for his help in computing the WCA parameters to the Nuida code.

This research was supported in part by the open collaborative research at National Institute of Informatics (NII) Japan (FY2018), and in part by JSPS KAKENHI Grants JP16K00185.

TABLE V
NUMBER OF CSCVS WITHIN RADIUS $d = 0.102$ MEASURED USING DYNAMIC METHOD AND COMPARISON OF COMPUTATIONAL COST AGAINST THAT OF SUBSET METHOD.

(a) $L = 1024$

$\boldsymbol{\theta_c^{str}}$	number c of colluders										
	2	3	4	5	6	7	8	9	10	$\bar{n}^{str}$	R^{str}
maj	4.93	1.55	1.60	3.70	3.76	4.79	4.86	3.41	3.47	3.56	0.40
min	4.93	2.15	1.11	1.01	1.00	1.14	1.21	1.32	1.31	1.69	0.19
coin	4.93	3.29	3.19	2.00	2.49	2.61	2.54	2.13	1.79	2.77	0.31
int	4.93	4.63	4.82	5.01	5.06	5.11	5.18	4.95	4.83	4.95	0.55
all0	1.00	1.01	1.22	1.87	2.43	3.14	3.50	3.26	2.52	2.22	0.25
all1	1.00	1.03	1.23	1.82	2.48	3.04	3.56	3.23	2.53	2.21	0.25
WCA	4.93	3.28	3.21	2.60	1.72	1.70	2.24	2.12	1.87	2.63	0.29

(b) $L = 2048$

$\boldsymbol{\theta_c^{str}}$	number c of colluders										
	2	3	4	5	6	7	8	9	10	$\bar{n}^{str}$	R^{str}
maj	5.44	1.60	1.46	3.42	3.47	5.06	5.10	3.22	3.24	3.56	0.40
min	5.44	2.86	1.05	1.00	1.02	1.19	1.26	1.52	1.55	1.88	0.21
coin	5.44	3.07	3.62	2.24	3.06	3.42	3.25	2.80	2.17	3.23	0.36
int	5.44	5.47	5.47	5.70	5.57	5.61	5.64	5.53	5.57	5.56	0.62
all0	1.00	1.00	1.17	1.78	2.48	3.11	3.49	3.25	2.51	2.20	0.24
all1	1.00	1.00	1.18	1.74	2.60	3.12	3.51	3.27	2.63	2.23	0.25
WCA	5.44	3.94	3.71	2.80	1.78	2.14	3.04	2.91	2.12	3.10	0.34

TABLE VI
COMPARISON OF SUM OF DETECTED COLLUDERS FOR $2 \leq c \leq 10$.

(a) $L = 1024$

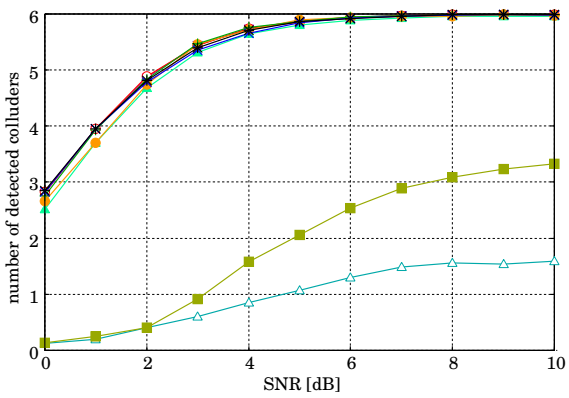
	maj	min	coin	int	all0	all1	WCA	total
MAP(optimal)	21.6	53.69	9.31	10.04	30.10	30.47	8.49	163.70
Symmetric [10]	7.14	6.17	6.88	6.87	6.55	6.82	6.79	47.22
WCA defence	9.84	10.99	9.00	8.79	8.82	9.07	8.47	64.98
Meerwald [13]	20.42	52.81	8.83	9.31	26.19	26.15	8.01	151.72
Bias Equalizer [24]	21.34	32.17	7.65	9.59	24.20	24.50	7.70	127.15
Basic Method	21.46	53.71	9.14	9.95	30.08	30.35	8.55	163.24
Subset Method	21.19	53.68	9.11	10.18	30.07	30.88	8.42	163.53
Dynamic Method	21.48	53.70	9.21	10.13	30.14	30.79	8.45	163.90

(b) $L = 2048$

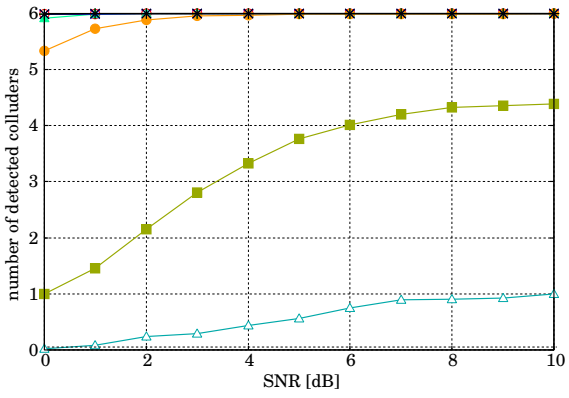
	maj	min	coin	int	all0	all1	WCA	total
MAP(optimal)	45.51	54.00	23.26	22.08	53.88	53.75	17.39	269.87
Symmetric [10]	14.62	13.14	13.82	14.45	13.86	14.02	14.23	98.14
WCA defense	19.17	22.95	20.24	19.03	19.78	20.05	17.32	138.54
Meerwald [13]	44.82	54.00	22.11	20.65	53.62	53.57	16.63	265.40
Bias Equalizer [24]	45.03	53.86	19.14	21.28	52.43	52.19	16.02	259.95
Basic Method	45.40	54.00	22.86	21.97	53.88	53.78	17.33	269.22
Subset Method	45.44	54.00	22.98	22.08	53.88	53.80	17.32	269.50
Dynamic Method	45.54	54.00	22.97	22.14	53.88	53.80	17.41	269.74

(c) $L = 2048$ in case of mix strategies

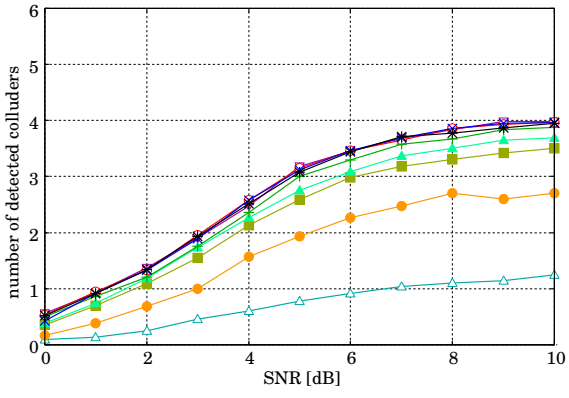
	mix1	mix2	mix3	total
MAP(optimal)	18.24	28.2	19.72	66.16
Symmetric [10]	14.06	14.70	14.24	43.00
WCA defence	17.56	18.63	19.44	55.63
Meerwald [13]	17.68	26.80	18.62	63.10
Bias Equalizer [24]	18.12	27.78	17.44	63.34
Basic Method	17.98	27.84	19.51	65.33
Subset Method	18.11	26.03	19.30	63.44
Dynamic Method	17.91	27.82	19.59	65.32



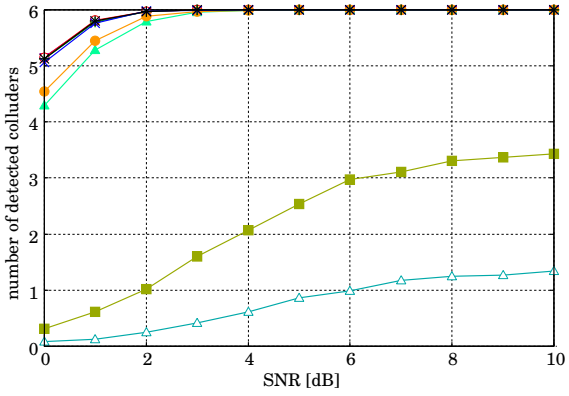
(a) majority



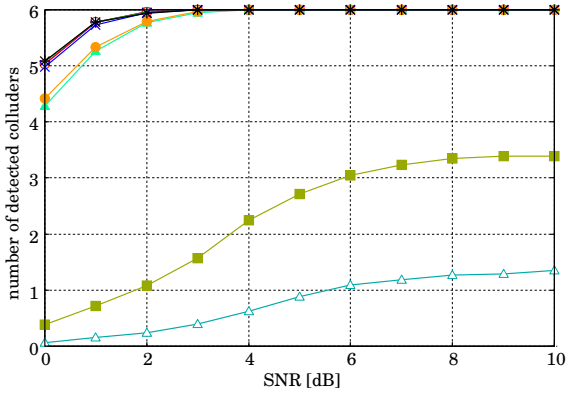
(b) minority



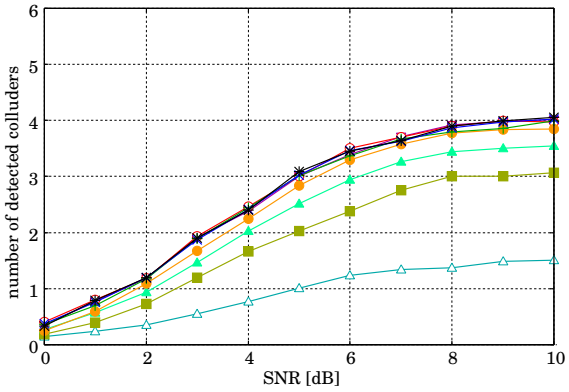
(c) coin-flip



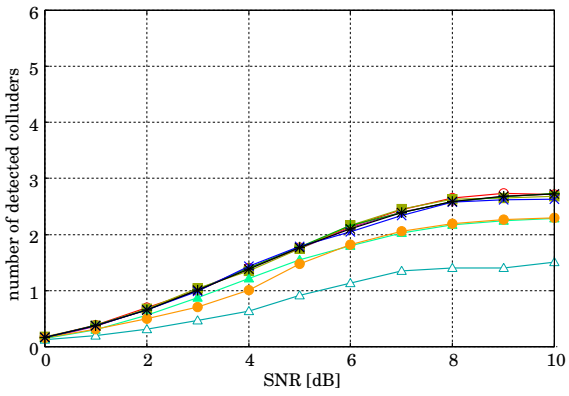
(d) all-0



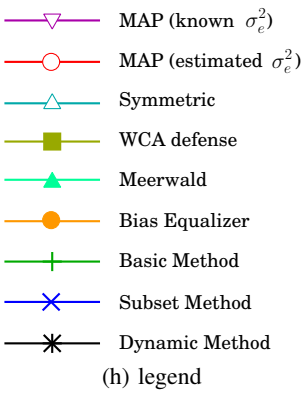
(e) all-1



(f) interleave



(g) WCA



(h) legend

Fig. 6. Comparison of traceability for typical collusion strategies.

TABLE VII
COMPARISON OF TOTAL DETECTED COLLUDERS IN NOISY CASE FOR SNR FROM 0–10 [DB].

	maj	min	coin	int	all0	all1	WCA	total
MAP (known σ_e^2) [11]	58.57	65.98	29.37	28.85	64.93	64.77	18.10	330.57
MAP (estimated σ_e^2) [11]	58.58	65.99	29.34	28.61	64.92	64.77	17.85	330.06
Symmetric [10]	10.73	6.10	7.76	10.02	8.36	8.54	9.44	60.95
WCA defence	20.42	35.80	24.80	20.42	24.34	25.12	17.94	168.84
Meerwald [13]	57.37	65.90	26.38	24.47	63.32	63.27	15.20	315.91
Bias Equalizer [24]	58.00	64.83	18.49	27.03	63.84	63.51	14.81	310.51
Basic Method	58.52	65.99	27.96	28.28	64.88	64.81	17.94	328.38
Subset Method	58.26	65.98	29.19	28.59	64.79	64.66	17.62	329.09
Dynamic Method	58.44	65.99	29.05	28.75	64.90	64.80	17.83	329.76

REFERENCES

[1] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inform. Theory*, vol. 44, no. 5, pp. 1897–1905, 1998.

[2] W. Trappe, M. Wu, Z. J. Wang, and K. J. R. Liu, "Anti-collusion fingerprinting for multimedia," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 1069–1087, 2003.

[3] Y. Yacobi, "Improved Boneh-Shaw content fingerprinting," in *Proc. CT-RSA 2001*. 2001, vol. 2020 of *LNCS*, pp. 378–391, Springer-Verlag.

[4] J. N. Staddon, D. R. Stinson, and R. Wei, "Combinatorial properties of frameproof and traceability codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 3, pp. 1042–1049, 2001.

[5] Y. T. Lin, J. L. Wu, and C. H. Huang, "Concatenated construction of traceability codes for multimedia fingerprinting," *Optical Engineering*, vol. 46, no. 10, pp. 107202.1–107202.15, 2007.

[6] G. Tardos, "Optimal probabilistic fingerprint codes," in *Proc. STOC 2003*, 2003, pp. 116–225.

[7] K. Nuida, M. Hagiwara, H. Watanabe, and H. Imai, "Optimization of Tardos's fingerprinting codes in a viewpoint of memory amount," in *Proc. IH 2007*. 2008, vol. 4567 of *LNCS*, pp. 279–293, Springer, Heidelberg.

[8] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, and H. Imai, "An improvement of discrete Tardos fingerprinting codes," *Designs, Codes and Cryptography*, vol. 52, no. 3, pp. 339–362, 2009.

[9] M. Wu, W. Trappe, Z. J. Wang, and K. J. R. Liu, "Collusion resistant fingerprinting for multimedia," *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 15–27, 2004.

[10] B. Škorić, S. Katzenbeisser, and M. Celik, "Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes," *Designs, Codes and Cryptography*, vol. 46, no. 2, pp. 137–166, 2008.

[11] T. Furon and L. Perez-Freire, "EM decoding of Tardos traitor tracing codes," in *ACM Multimedia and Security*, 2009, pp. 99–106.

[12] E. Abbe and L. Zheng, "Linear universal decoding for compound channels," *IEEE Trans. Inform. Theory*, vol. 56, no. 12, pp. 5999–6013, 2010.

[13] P. Meerwald and T. Furon, "Towards practical joint decoding of binary Tardos fingerprinting codes," *IEEE Trans. Inform. Forensics and Security*, vol. 7, no. 4, pp. 1168–1180, 2012.

[14] M. Desoubeaux, C. Herzet, W. Puech, and G. Le Guelvouit, "Enhanced blind decoding of Tardos codes with new MAP-based functions," in *Proc. MMSP*, 2013, pp. 283–288.

[15] J. J. Oosterwijk, B. Škorić, and J. Doumen, "A capacity-achieving simple decoder for bias-based traitor tracing schemes," *IEEE Trans. Inform. Theory*, vol. 61, no. 7, pp. 3882–3900, 2015.

[16] T Laarhoven, "Capacities and capacity-achieving decoders for various fingerprinting games," in *Proc. IH&MMSec2014*, 2014, pp. 123–134.

[17] T. Yasui, M. Kuribayashi, N. Funabiki, and I. Echizen, "Estimation of collusion attack in bias-based binary fingerprinting code," in *APSIPA 2018*, 2018, pp. 1550–1555.

[18] M. Kuribayashi, "Tardos's fingerprinting code over AWGN channel," in *IH 2010*. 2010, vol. 6387 of *LNCS*, pp. 103–117, Springer, Heidelberg.

[19] M. Kuribayashi, "Simplified MAP detector for binary fingerprinting code embedded by spread spectrum watermarking scheme," *IEEE Trans. Inform. Forensics and Security*, vol. 9, no. 4, pp. 610–623, 2014.

[20] T. Furon, L. P. Preire, A. Guyader, and F. Cérou, "Estimating the minimal length of Tardos code," in *IH 2009*. 2009, vol. 5806 of *LNCS*, pp. 176–190, Springer, Heidelberg.

[21] P. Moulin, "Universal fingerprinting: Capacity and random-coding exponents," *Proc. ISIT 2008*, pp. 220–224, 2008.

[22] A. Simone and B. Škorić, "Accusation probabilities in Tardos codes: beyond the gaussian approximation," *Designs, Codes and Cryptography*, vol. 63, no. 3, pp. 379–412, 2012.

[23] Frédéric Cérou, Pierre Del Moral, Teddy Furon, and Arnaud Guyader, "Sequential Monte Carlo for rare event estimation," *Statistics and Computing*, pp. 1–14, 2011.

[24] M. Kuribayashi and N. Funabiki, "Universal scoring function based on bias equalizer for bias-based fingerprinting codes," *IEICE Trans. Fundamentals*, vol. E101-A, no. 1, pp. 119–128, 2018.

PLACE PHOTO HERE

Tatsuya Yasui received B.E. degree from Okayama University, Japan, in 2018. He is currently pursuing the M.E. degree at the Graduate School of Natural Science and Technology, Okayama University. His current research interests include the digital fingerprinting code for the collusion-security.

PLACE PHOTO HERE

Minoru Kuribayashi received B.E., M.E., and D.E degrees from Kobe University, Japan, in 1999, 2001, and 2004. From 2002 to 2007, he was a Research Associate in the Department of Electrical and Electronic Engineering, Kobe University. In 2007, he was appointed as an Assistant Professor at the Division of Electrical and Electronic Engineering, Kobe University. Since 2015, he has been an Associate Professor in the Graduate School of Natural Science and Technology, Okayama University. His research interests include digital watermarking, information security, cryptography, and coding theory. He received the Young Professionals Award from IEEE Kansai Section in 2014.

PLACE PHOTO HERE

Nobuo Funabiki received the B.S. and Ph.D. degrees in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984 and 1993, respectively. He received the M.S. degree in electrical engineering from Case Western Reserve University, USA, in 1991. From 1984 to 1994, he was with the System Engineering Division, Sumitomo Metal Industries, Ltd., Japan. In 1994, he joined the Department of Information and Computer Sciences at Osaka University, Japan, as an assistant professor, and became an associate professor in 1995. He stayed at University of California, Santa Barbara, in 2000-2001, as a visiting researcher. In 2001, he moved to the Department of Communication Network Engineering (currently, Department of Electrical and Communication Engineering) at Okayama University as a professor. His research interests include computer networks, optimization algorithms, educational technology, and Web technology. He is a member of IEEE, IEICE, and IPSJ. He is the associate editor-in-chief in Journal of Communications since 2016 and an editor in International Journal of Computer & Software Engineering since 2018. He was the chairman at IEEE Hiroshima section in 2015 and 2016. He has served as a member of technical program committees in more than 30 international conferences. He has published more than 600 papers and coauthored seven books.

PLACE PHOTO HERE

Isao Echizen received B.S., M.S., and D.E. degrees from the Tokyo Institute of Technology, Japan, in 1995, 1997, and 2003, respectively. He joined Hitachi, Ltd. in 1997, and until 2007 was a research engineer in the company's systems development laboratory. He is currently a deputy director general of the National Institute of Informatics (NII), a professor of the Information and Society Research Division, the NII, and a professor in the Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo. He is also a visiting professor at the Tsuda University and was a visiting professor at the University of Freiburg in 2010 and at the University of Halle-Wittenberg in 2011. He has been engaged in research on information security and content security and privacy. He received the Best Paper Award from the IPSJ in 2005 and 2014, the Fujio Frontier Award and the Image Electronics Technology Award in 2010, the One of the Best Papers Award from the Information Security and Privacy Conference and the IPSJ Nagao Special Researcher Award in 2011, the Docomo Mobile Science Award in 2014, the Information Security Cultural Award in 2016, and the Best Paper Award at the IEEE WIFS 2017.He is a member of the Information Forensics and Security Technical Committee and the IEEE Signal Processing Society.