Estimation of Collusion Attack in Bias-based Binary Fingerprinting Code

Tatsuya Yasui*, Minoru Kuribayashi*, Nobuo Funabiki*, and Isao Echizen†
* Okayama University, Okayama, Japan
E-mail: {yasui.tatsuya@s., kminoru@, funabiki@}okayama-u.ac.jp
† National Institute of Informatics, Tokyo, Japan
E-mail: iechizen@nii.ac.jp

***Abstract*—An optimal detector known as MAP detector has been proposed for the probabilistic fingerprinting codes such as Tardos and Nuida codes. However, it needs two kinds of important information. One is the collusion strategy which is used at the generation of a pirated codeword from colluders' codewords, and the other is the number of colluders. In this study, we propose an estimator which outputs these two parameters from a pirated codeword. At the estimation, we measure a bias in the pirated codeword by observing the number of symbols "0" and "1", and compare with possible bias patterns calculated from collusion strategies and number of colluders. As a result of computer simulation, it is confirmed that a collusion strategy and number of colluders can be estimated with high probability. In addition, it is revealed that the traceability of the detector using the proposed estimator is extremely close to the optimal detector.**

I. Introduction

In the collusion-secure codes [1], Tardos code [2] is known as bias-based fingerprinting code such that each symbol of a codeword is determined by a certain biased probabilistic distribution. As the code length has been proven to be theoretically minimum order, the Tardos code has been intensively investigated to further improve the performance in the viewpoints of traceability as well as the code length. In particular, Nuida et al. [3], [4] constructed an interesting variants using a discrete probabilistic distribution(Gauss-Legendre distribution) to customize the bias-based fingerprinting code for a fixed number of possible colluders. For convenience, this fingerprinting code is called Nuida code in this paper.

In order to identify illegal users called colluders from a pirated codeword, a tracing algorithm called detector finds suspicious users by calculating similarity with their codewords. The detector can be classified into 3 types: catch-one, catch-many, and catch-all [5]. In case of catch-one, the most suspicious user whose similarity score becomes maximum is detected as guilty. As we assume a collusion of some illegal users, a catch-many type detector is desirable because it can identify as many illegal users as possible. Although all colluders can be identified in case of catch-all, its false-negative rate that no colluder is detected must be higher. Therefore, we focus on the catch-many type detector in this study.

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. One is the scoring function which calculates similarity scores, and the other is the classification using a threshold. Even if colluders can choose an arbitrary collusion strategy such as majority and minority voting to generate a pirated codeword from their codewords, the Tardos's and its revised scoring functions [6] are independent on the collusion strategy, and hence, their performance is not high. According to an information theoretical analysis, an optimal detector which can calculates a best score has been proposed by Furon et al. [7] to utilize the information about the collusion strategy and the number of colluders. Because of the difficulty at the estimation of these parameters [7], some researchers [8], [9], [10], [11] investigated a defense strategy to minimize the performance gap from the optimal detector.

In this study, we proposed an effective estimator for these parameters by using the characteristics of discretized probabilistic distribution of Nuida code. This estimator is composed of two steps. At the first step, we observe the bias of symbol "1" in a pirated codeword and make a feature vector according to the characteristics of bias-based fingerprinting code. Essentially, each symbol of codeword is determined by each assigned bias probability. Therefore, the bias of symbol "1" in each symbol of innocent users' codewords is statistically stable and depend only on the bias probability. On the other hand, the bias in a pirated codeword is different and is affected by the collusion strategy and number of colluders. Because the candidates for the bias probability in Nuida code is finite, we classify the symbols in a pirated codeword into groups having same bias probabilities. Then, we calculate each expected probability that the symbols in a group become "1" after a collusion attack. For each collusion strategy and number of colluders, the set of the expected probabilities are different. For convenience, such a set is defined as a Collusion Strategy Characteristic Vector (CSCV). At the second step, we try to find the closest CSCV which distance from the feature vector becomes minimum to estimate the collusion strategy and number of colluders. As a result of this study, the estimation accuracy becomes more than 90 percent in well-known 7 collusion strategies and the traceability is fairly close to the optimal one.

TABLE I
EXAMPLE OF THE 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			

II. FINGERPRINTING CODE

This section reviews how to construct the bias-based fingerprinting code and how to design its tracing algorithm. We also show several collusion strategies to generate a pirated codeword.

A. Bias-based Binary Fingerprinting Code

The Tardos code [2] is a binary bias-based fingerprinting code composed of N codewords with L symbols. Let $\boldsymbol{x_j} = (x_{j,1}, \ldots, x_{j,i}, \ldots, x_{j,L})$ be a codeword of j-th user, where $x_{j,i} \in \{0,1\}(1 \le i \le L)$ is generated from an independently and identically distributed random number with a probability p_i such that $\Pr[x_{j,i} = 1] = p_i$ and $\Pr[x_{j,i} = 0] = 1 - p_i$. The probability $p_i \in \boldsymbol{P}$ follows a certain continuous distribution over an open unit interval (0,1) called bias distribution.

For the improvement of the performance of Tardos code, Nuida et al. [3], [4] presented a discrete version of the bias distribution, which is customized for a given the maximum number c_{max} of colluders. Let $L_k(t) = (\frac{d}{dt})^k (t^2-1)^k/(k!2^k)$ be the k-th Legendre polynomial, and put $\tilde{L}_k(t) = L_k(2t-1)$. Then we define $\mathcal{P}^{GL}_{2k-1} = \mathcal{P}^{GL}_{2k}$ to be the finite probability distribution whose values are the k zeroes of $\tilde{L}_k$, with each value p taken with probability $\eta(p(1-p))^{-3/2}\tilde{L}'_k(p)^{-2}$, where η is the normalized constant making the sum of the probabilities equal to 1.

Similar to the Tardos code, the codewords of Nuida code is generated by 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 $n_g = \lceil c_{max}/2 \rceil$. The numerical examples are shown in Table I, where P_ξ and Q_ξ, for $1 \le \xi \le n_g$ respectively denote the values of discretized probabilities and their emerging probabilities. For example, when $c_{max} = 8$ and the length of the sequence $\boldsymbol{P}$ is $L = 10000$, the number of elements which has $p_i = P_2 = 0.33001$ is approximately $L \cdot Q_2 \approx 2517$ in average. Since each symbol $x_{j,i}$ of users' codewords is independently and identically selected under the constraint of $\Pr[x_{j,i} = 1] = p_i$, symbols of a codeword $\boldsymbol{x_j}$ can be separated into n_g groups based on $p_i \in \boldsymbol{P}$.

B. Collusion attack

Suppose that c colluders attempt to produce a pirated copy from their fingerprinting codes. Under the marking assumption [1], a pirated codeword $\boldsymbol{y} = \{y_1, y_2, \ldots, y_L\}$ $y_i (1 \le i \le L)$ is constructed with 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 yield the pirated sequence $\boldsymbol{y}$ as an output. In case fingerprinting codes were attacked, 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 index in $\boldsymbol{I_i}$ is identical because their positions are undetectable.

In [12], the collusion attack is defined by the parameter vector $\boldsymbol{\theta^{str}_c} = (\theta^{str}_0, \ldots, \theta^{str}_c)$ with $\theta^{str}_\lambda = \Pr[y_i = 1 | \Phi = \lambda](0 \le \lambda \le c)$, where $\Phi \in \{0, \ldots, c\}$ denotes the number of symbol "1" in the colluders' copies at a given index. It is reported in [12] that some collusion strategies have a deeper impact on the traceability than others and Worst Case Attack (WCA) which minimizes the achievable rate of the code is defined from information theoretical point of view. The marking assumption enforces that $\theta^{str}_0 = 0$ and $\theta^{str}_c = 1$ in the collusion strategies. In case of $c = 6$, some typical examples 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:
  $\boldsymbol{\theta^{all-0}_6}$ = (0,0,0,0,0,0,1).
- All-1:
  $\boldsymbol{\theta^{all-1}_6}$ = (0,1,1,1,1,1,1).
- Interleave:
  $\boldsymbol{\theta^{int}_6}$ = (0,$\frac{1}{6}$,$\frac{2}{6}$,$\frac{3}{6}$,$\frac{4}{6}$,$\frac{5}{6}$,1).
- WCA:
  $\boldsymbol{\theta^{WCA}_6}$ = (0,0.5014,0.1749,0.5,0.8251,0.4986,1).

C. Tracing algorithm

A tracing algorithm called detector is composed of a scoring function and a classification. We should consider the following two error rates ϵ_{FP} and ϵ_{FN}, where the tracing algorithm Tr outputs suspicious users. ϵ_{FP} : false-positive

$$\epsilon_{FP} = \Pr[Tr(\boldsymbol{y}) \notin \boldsymbol{C} | Tr(\boldsymbol{y}) \neq \emptyset].$$

ϵ_{FN} : false-negative

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

Tardos proposed the following scoring function.

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

where

$$U_{j,i} = \begin{cases} -\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i} = 1), \\ \sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i} = 0). \end{cases} \tag{2}$$

At the classification, only one suspicious user whose score becomes maximum is determined as an illegal user in catch-one type. The scoring function in Eq. (1) can be applied for the

Nuida code. Unfortunately, the scoring function only uses half of information about a pirated codeword because the value of the score $S_{j,i}$ becomes zero when $y_i = 0$. In order to utilize the entire information, Škorić et al. [6] proposed the following symmetric version of the scoring function.

$$S_j^{sym} = \sum_{i=1}^{L} S_{j,i}^{sym} = \sum_{i=1}^{L} (2y_i - 1)U_{j,i}, \tag{3}$$

The above scoring functions need no information about the collusion strategy $\boldsymbol{\theta_c^{str}}$ and the number c of colluders. In order to discriminate colluders from innocents, an optimal scoring function should be designed by using these parameters from the information theoretical point of view. In [7], the optimal scoring function is given by the following log-likelihood ratio,

$$S_j^{MAP} = \sum_{i=1}^{L} S_{j,i}^{MAP} = \sum_{i=1}^{L} \log\left(\frac{\Pr[y_i|x_{j,i}, \boldsymbol{\theta_c^{str}}]}{\Pr[y_i|\boldsymbol{\theta_c^{str}}]}\right). \tag{4}$$

As the above score calculates the maximum a posteriori probability, the optimal scoring function is called MAP detector. The denominator $\Pr[y_i|p_i, \boldsymbol{\theta_c^{str}}]$ can be calculated by

$$\begin{cases} \Pr[1|\boldsymbol{\theta_c^{str}}] = \sum_{\rho=0}^{c} \theta_\rho^{str} \binom{c}{\rho} p_i^\rho (1-p_i)^{c-\rho}, \\ \Pr[0|\boldsymbol{\theta_c^{str}}] = 1 - \Pr[1|\boldsymbol{\theta_c^{str}}]. \end{cases} \tag{5}$$

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

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

The difficulty in designing such an optimal scoring function is how to estimate the collusion strategy $\boldsymbol{\theta_c^{str}}$ and the number c of colluders from a given codeword $\boldsymbol{y}$. Although, these parameters are estimated by using an Expectation-Maximization(EM) algorithm in [7], its accuracy is not high. To the best of our knowledge, there is no other study to investigate the estimator.

D. Threshold

Some suspicious users whose score exceeds a threshold Z are regarded as illegal users in a catch-many type detector. Some methods approximate the distribution of user's score S_j by the Gaussian distribution [14] to calculate the threshold for satisfying a given false-positive probability. With the increase of the length of users' codewords, we can approximate it more accurately. However, it is reported in [15] that such an approximation is not appropriate for the calculation of threshold to suppress the false-positive rate to be less than ϵ_{FP} because the tail part of the Gaussian distribution is not accurate with the short length of codeword. For the purpose of accurate measurement the tail part, Furon et al. [12] proposed an efficient method for estimating the probability of rare events, which is called rare event simulator. We can estimate the ϵ_{FP} for a given threshold Z by this method, which means that we calculate the mapping $\epsilon_{FP} = F(Z)$.

III. Conventional study

We discuss the universal scoring function that the performance for arbitrary collusion strategy is higher than the uninformed scoring function like Škorić's symmetric scoring function. Because of the difficulty of realization of MAP detector, the scoring function has been adjusted for a certain collusion strategy to achieve the universality [8], [9], [10], [11]. On the other hand, the bias of symbols "0" and "1" is observed and the score is calculated introducing the weights corresponding to biases in the Škorić's scoring function in [13]. In this section, we review two examples of scoring function for our proposed estimator.

A. Scoring Function Based on MAP

A simple conversion from the MAP detector is to fix the collusion strategy $\boldsymbol{\theta_c^{str}}$ at the scoring function. In [8], under an assumption that the actual number of colluders c is less than or equal to c_{max}, Meerwald et al. [8] calculated the correlation scores for c selected from $[1, c_{max}]$, where the scoring function is based on the MAP detector designed for WCA $\boldsymbol{\theta_t^{WCA}}(1 \le t \le c_{max})$. The score $S_{j,i}^{Mee}$ is determined by one of the c_{max} candidates which value becomes maximum.

$$S_j^{Mee} = \sum_{i=1}^{L} \max_{1 \le t \le c_{max}} \left(\log \frac{\Pr[y_i|x_{j,i}, \boldsymbol{\theta_t^{WCA}}]}{\Pr[y_i|\boldsymbol{\theta_t^{WCA}}]}\right). \tag{7}$$

Similarly, Desoubeaux [9] designed the scoring function for coin-flip attack defense and summed c_{max} candidates

$$S_j^{Des} = \sum_{i=1}^{L} \log\left(\sum_{t=1}^{c_{max}} t \cdot \left(\frac{\Pr[y_i|x_{j,i}, \boldsymbol{\theta_t^{coin}}]}{\Pr[y_i|\boldsymbol{\theta_t^{coin}}]}\right)\right). \tag{8}$$

B. Bias Equalizer

In binary fingerprinting codes, the number of symbols "0" and "1" is balanced because of the symmetry of bias probability p_i. However, it is not always balanced in a pirated codeword. Considering the imbalance caused by a collusion attack, the Skoric's scoring function is revised by equalizing the balance using weighting parameters in [13], which is called Bias Equalizer. 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 number 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 to be $L_1 = L_0$ unless colluders do not know the actual values $x_{j,i}$ of their codewords. Therefore, in case of $\boldsymbol{y}$ produced by "all-0" and "all-1", L_1 is not always equal to L_0 in a real situation. The number of elements in ξ-th group is denoted by ℓ_ξ, where $\ell_\xi \ge 0$ and $\sum_{\xi=1}^{n_g} \ell_\xi = L$. Besides, the number of symbols "1" and "0" are denoted by $\ell_{\xi,1}$ and $\ell_{\xi,0}$, respectively. Notice 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

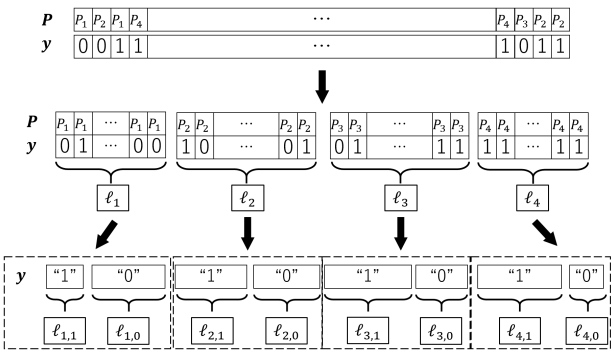

Fig. 1. Number of symbols "0" and "1" in a pirated codeword.

in Fig.1. Using those parameters, the scoring function in Bias Equalizer is showed by the following score.

$$S^{Bias}_{j,i,\xi} = y_i \begin{cases} U^{00}_{j,i} = \dfrac{\ell_{\xi,1}}{\ell_\xi}\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i}=y_i=0), \\ U^{01}_{j,i} = -\dfrac{\ell_{\xi,0}}{\ell_\xi}\sqrt{\dfrac{p_i}{1-p_i}} & (x_{j,i}=1, y_i=0), \\ U^{10}_{j,i} = -\dfrac{\ell_{\xi,1}}{\ell_\xi}\sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i}=0, y_i=1), \\ U^{11}_{j,i} = \dfrac{\ell_{\xi,0}}{\ell_\xi}\sqrt{\dfrac{1-p_i}{p_i}} & (x_{j,i}=y_i=1). \end{cases} \tag{9}$$

In order to adjust the above weighting parameters according to the gap for the all-0, all-1, minority, and coin-flip attacks, the collusion strategy by the following conditions observed from $\boldsymbol{y}$ which all-0 or all-1 attack is performed.

$$\begin{cases} \ell_{\xi,0} \approx \ell_\xi, & \text{if } p_i < 0.5 \text{ holds for all } \xi \\ \ell_{\xi,1} \approx \ell_\xi, & \text{if } p_i > 0.5 \text{ holds for all } \xi \end{cases} \tag{10}$$

For the classification of all-0 and all-1 attack, a threshold $T^\dagger$ is used to check the following two cases:

$$\begin{cases} \dfrac{\ell_{\xi,0}}{\ell_\xi} > T^\dagger & (p_i < 0.5), \\ \dfrac{\ell_{\xi,1}}{\ell_\xi} > T^\dagger & (p_i > 0.5). \end{cases} \tag{11}$$

Note that $T^\dagger$ is close to 1 because of relation given by Eq.(10). In the paper [13], the threshold $T^\dagger$ was empirically determined: $T^\dagger = 0.95$. When the minority or coin-flip attack strategy is performed, the following relations can be observed for ξ-th group.

$$\begin{cases} \dfrac{\ell_{\xi,0}}{\ell_{\xi,1}} < \sqrt{\dfrac{1-p_i}{p_i}} & (p_i < 0.5), \\ \dfrac{\ell_{\xi,1}}{\ell_{\xi,0}} < \sqrt{\dfrac{p_i}{1-p_i}} & (p_i > 0.5). \end{cases} \tag{12}$$

Even though the Bias Equalizer improves the performance of scoring function, the classification of collusion strategies was not theoretically investigated.

TABLE II
VALUE OF COLLUSION STRATEGY CHARACTERISTIC VECTORS IN CASE OF $c = 6$.

$\boldsymbol{\Gamma}^{\boldsymbol{str}}_{\boldsymbol{6}}$		ξ			
		$\gamma^{str}_{6,1}$	$\gamma^{str}_{6,2}$	$\gamma^{str}_{6,3}$	$\gamma^{str}_{6,4}$
str	$\boldsymbol{\Gamma}^{\boldsymbol{maj}}_{\boldsymbol{6}}$	0.0030	0.2050	0.7950	0.9970
	$\boldsymbol{\Gamma}^{\boldsymbol{min}}_{\boldsymbol{6}}$	0.3476	0.7059	0.2941	0.6524
	$\boldsymbol{\Gamma}^{\boldsymbol{coin}}_{\boldsymbol{6}}$	0.1753	0.4554	0.5446	0.8247
	$\boldsymbol{\Gamma}^{\boldsymbol{all0}}_{\boldsymbol{6}}$	0.0000	0.0013	0.0905	0.6494
	$\boldsymbol{\Gamma}^{\boldsymbol{all1}}_{\boldsymbol{6}}$	0.3506	0.9096	0.9987	1.0000
	$\boldsymbol{\Gamma}^{\boldsymbol{int}}_{\boldsymbol{6}}$	0.0694	0.3300	0.6700	0.9306
	$\boldsymbol{\Gamma}^{\boldsymbol{WCA}}_{\boldsymbol{6}}$	0.1581	0.3747	0.6253	0.8418

IV. Proposed Estimator

This section shows how to estimate the collusion strategy $\boldsymbol{\theta}^{\boldsymbol{str}}_{\boldsymbol{c}}$ and the number c of colluders for the optimal detector (MAP). We exploit the bias in a pirated codeword at the estimation.

A. Collusion Strategy Characteristic Vector (CSCV)

When a pirated codeword is produced by a combination of some codewords under the constraint of marking assumption, the number of symbols "0" and "1" must be changed. We measure the amount of changes according to the discrete 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:

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

On the other hand, the right term in Eq.(13) will be changed in a pirated codeword and the amount 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

$$\boldsymbol{\Gamma} = (\gamma_1, \ldots, \gamma_\xi, \ldots, \gamma_{n_g}), \text{where } \gamma_\xi = \ell_{\xi,1}/\ell_\xi. \tag{14}$$

The expectation of elements in $\boldsymbol{\Gamma}$ can be calculated from θ^{str}_c and c:

$$\gamma^{str}_{c,\xi} = \sum_{t=1}^{c} \binom{c}{t} P^t_\xi (1-P_\xi)^{c-t} \theta^{str}_t \tag{15}$$

The vector $\boldsymbol{\Gamma}^{\boldsymbol{str}}_{\boldsymbol{c}} = (\gamma^{str}_{c,1}, \ldots, \gamma^{str}_{c,\xi}, \ldots, \gamma^{str}_{c,n_g})$ is called Collusion Strategy Characteristic Vector (CSCV). Under the marking assumption, Eq.(15) enables us to express $\boldsymbol{\Gamma}^{\boldsymbol{str}}_{\boldsymbol{c}}$ of every general collusion strategy we can conceive. Some examples for typical collusion strategies are shown in Table II, where $c_{max} = 8$ for Nuida code and the actual number of colluders is $c = 6$. Different from these thresholds, we measure the distance from the feature vector $\boldsymbol{\Gamma}$ to find the closest $\boldsymbol{\Gamma}^{\boldsymbol{str}}_{\boldsymbol{c}}$ for possible collusion strategies.

TABLE III
ACCURACY OF ESTIMATOR IN A BASIC METHOD WHEN c IS KNOWN.

(a) $D_1^{str,c}$

$D_1^{str,c}$		number c of colluders						
		2	3	4	5	6	7	8
$\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$	maj	100	100	100	100	100	100	100
	min	0	99.8	100	100	100	100	100
	coin	0	90.3	55.7	97	99.5	100	100
	int	0	97.2	100	100	100	100	100
	all0	100	100	100	100	100	100	100
	all1	100	100	100	100	100	100	100
	WCA	0	91.3	57.4	95.7	99.7	100	100

(b) $D_2^{str,c}$

$D_2^{str,c}$		The number c of colluders						
		2	3	4	5	6	7	8
$\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$	maj	100	100	100	100	100	100	100
	min	0	99.8	100	100	100	100	100
	coin	0	89.4	57.7	96.8	99.7	100	100
	int	0	96.8	99.9	100	100	100	100
	all0	100	100	100	100	100	100	100
	all1	100	100	100	100	100	100	100
	WCA	0	93.4	57.8	95.8	99.8	100	100

TABLE IV
COMPARISON OF VALUES WHICH CALCULATED BY ESTIMATOR IN CASE OF $c = 4$(MAJORITY, COIN-FLIP, WCA).

str	$\gamma_{4,1}$	$\gamma_{4,2}$	$\gamma_{4,3}$	$\gamma_{4,4}$
majority	0.013792	0.254839	0.745161	0.986208
coin-flip	0.125068	0.405181	0.594819	0.874932
WCA	0.122175	0.401272	0.598728	0.877825

B. Basic Method

Let $D^{str,c}$ be the distance between the observed vector $\boldsymbol{\Gamma}$ and the CSCV $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$. Notice that $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}$ can be calculated in advance and can be stored at a database. In the classification of collusion strategy and number of colluders, we calculates $D^{str,c}$ for all strategies $\boldsymbol{\theta}_{\boldsymbol{c}}^{\boldsymbol{str}}$, and find the one which $D^{str,c}$ becomes minimum.

$$\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}} = \arg\min_{str,c} D^{str,c}. \tag{16}$$

Well-known metrics for distance are the Manhattan distance and the Euclidean distance:

$$D_1^{str,c} = \sum_{\xi} |\gamma_{c,\xi}^{str} - \gamma_{\xi}|, \tag{17}$$

$$D_2^{str,c} = \sqrt{\sum_{\xi} \left(\gamma_{c,\xi}^{str} - \gamma_{\xi}\right)^2}. \tag{18}$$

We store $\boldsymbol{\Gamma}_{\boldsymbol{c}}^{\boldsymbol{str}}(c_{min} \le c \le c'_{max})$ with general collusion strategy into the database, where c_{min} and c'_{max} are the minimum and maximum number of colluders that we assume, respectively.

C. Dynamic Method

The proposed estimator searches possible collusion attacks by using CSCVs stored in a database, namely it is a kind of exhaustive search. Instead of the exhaustive search, we proposed a dynamic estimation method based on the idea in [8], which calculates a set of user's scores for some candidate number of colluders and outputs the one which score becomes maximum. We first estimate the strategy $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$ for a given c ($c_{min} \le c \le c'_{max}$) by Eq.(16). Then, with all estimated strategy $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$ for c, we calculate the score $S_{j,i}$ and select the maximum score. Finally, the total score S_j is obtained by the summation of them. The following process is how to determine the user j-th score in this method.

1) Initialize $c = c_{min}$.
2) Input c and estimate $\boldsymbol{\theta}_{\boldsymbol{c}}^{\tilde{\boldsymbol{str}}}$ in Eq.(16).
3) Increment $c = c + 1$.
4) If $c = c'_{max}$, go to next process; otherwise go to process 1.
5) Calculate the score S_j by the following equation.

$$S_j = \sum_{i=1}^{L} \max_{c_{min} \le t \le c'_{max}} \left(\log \frac{\Pr[y_i | x_{j,i}, \boldsymbol{\theta}_{\boldsymbol{t}}^{\tilde{\boldsymbol{str}}}]}{\Pr[y_i | \boldsymbol{\theta}_{\boldsymbol{t}}^{\tilde{\boldsymbol{str}}}]} \right). \tag{19}$$

For convenience, the estimator presented in Section IV-B is called basic method, and the dynamic estimator explained above is called dynamic method.

V. EXPERIMENTAL RESULTS

In this section, for the comparison of the performance of proposed two methods, we perform a simulation. The experimental setup is the followings. The number of users in a system is $N = 10^6$, and a codeword of length $L = 2048$ is assigned to a user. We use the Nuida code designed by $c_{max} = 8$ and the false-positive probability is fixed to be $\epsilon = 10^{-10}$ by using a rare event simulator [15]. The candidates of collusion strategies are $str = \{$majority, minority, coin-flip, all-0, all-1, interleave, WCA$\}$, and the number of colluders ranges from $c_{min} = 2$ to $c'_{max} = 10$. Pirated codewords are produced by collusion attack on randomly selected 10^3 combinations of c colluders.

Assuming that the number c of colluders is known in advance, the accuracy of estimator in basic method is measured. Table III shows the accuracy of estimated collusion strategies calculated by $D_1^{str,c}$ and $D_2^{str,c}$ distances for $2 \le c \le 8$, respectively. The more the number of colluders increase, the higher accuracy of estimation is. However, in case of $c = 4$ and coin-flip or WCA strategy, the accuracy is dropped because of the similarity between $\boldsymbol{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{coin}}$ and $\boldsymbol{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{WCA}}$. Table IV shows that the comparison of $\boldsymbol{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{maj}}$, $\boldsymbol{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{coin}}$ and $\boldsymbol{\Gamma}_{\boldsymbol{4}}^{\boldsymbol{WCA}}$. It is known from the preliminary experiment that the impact of misestimation of coin-flip and WCA is much less than any other strategies. We determined to use the $D_1^{str,c}$ because the impact of coin-fips is much less than WCA one.

Table V shows the sum of detected colluders for $2 \le c \le 10$, where the maximum is 54 = $\sum_{c=2}^{10} c$. In case of MAP detector,

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

	majority	minority	coin-flip	interleave	all-0	all-1	WCA	total
Symmetric [6]	14.67	13.31	13.87	14.33	13.88	13.92	14.01	97.99
MAP(optimal)	45.31	54.00	23.14	21.87	53.85	53.84	17.98	269.98
Basic Method	45.31	54.00	22.73	21.74	53.85	53.85	17.86	269.33
Dynamic Method	45.23	54.00	22.97	21.88	53.86	53.87	17.95	269.74
Meerwald [8]	18.91	23.07	20.32	18.67	20.09	20.14	17.94	139.14
Bias Equalizer [13]	45.11	53.81	19.16	21.34	52.39	52.30	16.41	260.52

the collusion strategy and the number of colluders are informed, and hence, the number of detected colluders in MAP is the theoretical upper limit. It is observed that the result exceeds the limit for all-1 attack in the proposed methods. It is because of the probabilistic algorithm in the rare event simulator [15] at the calculation of the threshold. If the number of trials is increased, such a case will not be occurred. It is observed that the traceability of basic and dynamic method is very close to the optimal detector (MAP) for all collusion strategies. In the comparison of proposed two methods, the dynamic method is better than the basic method without majority attack.

VI. Conclusion

In this paper, we proposed an estimator of collusion attack for optimal detector (MAP) using the imbalance of symbols in a pirated codeword. According to the discrete bias probabilities in the Nuida code, we classify the imbalances into groups. The imbalances are calculated for possible collusion strategies and number of colluders as the CSCV. At the estimation, the distances between the imbalances observed from a pirated codeword and CSCVs stored in a database are examined to find the closest one. As a result of computer simulation, it was confirmed that the total performance of proposed method was better than any other conventional studies and was very close to the MAP performance. One of our future work is to consider the additive noise in a pirated codeword.

Acknowledgment

The research is supported by the open collaborative research at National Institute of Informatics (NII) Japan (FY2018).

References

[1] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," IEEE Trans. Inform. Theory, vol.44, pp.1897–1905, 1998.
[2] G. Tardos, "Optimal probabilistic fingerprint codes," Proc. STOC 2003, pp.116–225, 2003.
[3] K. Nuida, M. Hagiwara, H. Watanabe, and H. Imai, "Optimization of Tardos's fingerprinting codes in a viewpoint of memory amount," Proc. IH 2007, LNCS, vol.4567, pp.279–293, Springer, Heidelberg, 2008.
[4] K. Nuida, S. Fujitu, 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.
[5] 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. 1527, 2004.
[6] 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.
[7] T. Furon and L.P. Freire "EM decording of Tardos traitor tracing codes," ACM Multimedia and Security, pp.99–106, 2009.
[8] 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.
[9] M. Desoubeaux, C. Herzet, W. Puech, and G. L. Guelvouit, "Enhanced blind decoding of Tardos codes with new MAP-based functions," Proc. MMSP, pp.283–288, 2013.
[10] J. J. Oosterwijk, B. Skorikc, 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.
[11] T. Laarhoven, "Capacities and capacity-achieving decoders for various fingerprinting games," Proc. IH&MMSec2014, pp.123–134, 2014.
[12] T. Furon, L. P. Preire, A. Guyader, and F. Cérou, "Estimating the minimal length of Tardos code," Proc. IH 2009, LNCS, vol.5806, pp.176–190, Springer, Heidelberg, 2009.
[13] 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.
[14] M. Kuribayashi, "Tardos's fingerprinting code over AWGN channel," Prof. IH2010, LNCS, vol. 6387. Springer, Heidelberg, pp.103–117, 2010.
[15] 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.