Regular Paper

Random-Error-Resilient Tracing Algorithm for a Collusion-Secure Fingerprinting Code

Katsunari Yoshioka† and Tsutomu Matsumoto†

Many collusion-secure fingerprinting codes drop their performance of traitor tracing when random errors are added on the embedded data. We propose a new tracing algorithm of *c-secure CRT code*, one of the collusion-secure fingerprinting codes. This algorithm enhances the resilience against the random-error addition by introducing a threshold for distinguishing *detected blocks*, part of the embedded data altered by the collusion attack. Based on numerical results by computer simulations, this threshold is proved to be practical in decreasing the false tracing rate in the case that both the collusion attack and the random-error addition are done.

1. Introduction

For the sake of copyright protection of digital data such as images, videos, audios, texts and software programs, a distinct ID is embedded in each copy of the content as a *digital watermark*. When a pirated copy is found, the traitor is traced from the ID extracted from the copy. This application of a watermarking scheme is called *fingerprinting* and the ID embedded into the content is called a *fingerprint*.

Traitors that make alterations on the content in order to avoid the tracing are called attackers. The goal of the attackers is to remove or destroy fingerprints without giving major damage to the content from the alteration they make. Collusion attacks are conducted by attackers that have obtained two or more fingerprinted contents. By comparing the obtained contents, they are able to detect some locations of the embedded data and alter them without worrying about damaging the content. With collusion attacks, it becomes more likely for the attackers to generate a copy assigned to users that are not in the coalition (i.e., framing innocent users).

To maintain the resilience against collusion attacks, the use of a collusion-secure fingerprinting code is considered. The collusion-secure fingerprinting code shown by Boneh and Shaw[1] is called *c-secure code with error* ϵ, which prevents any coalitions consisting of c attackers or less to generate a copy that the tracer cannot identify even one of the traitors from. Then, shorter versions of c-secure code are proposed[5),8),11] to reduce the huge length of the c-secure code with error ϵ.

However, the tracing algorithms for these collusion-secure fingerprinting codes above are designed assuming the collusion attacks are the only possible attacks against the scheme. Therefore, they drop their performance of traitor tracing when altered by attacks such as random error addition. As it is quite easy for the attackers to add random errors into the embedded data, it is necessary to improve the random-error-resilience of the collusion-secure fingerprinting code. Examples of these attacks are compression, scale change, conversion between data formats and cropping against digital watermarkings on images.

We introduce a random-error-resilient tracing algorithm for *c-secure CRT code*[5], one of the collusion-secure fingerprinting codes. In Section 2, the fingerprinting scheme and the attacks against the scheme are described. Also, the properties of collusion-secure fingerprinting codes are explained. In Section 3, there will be an explanation on the c-secure CRT code. Then, we propose the random-error-resilient tracing algorithm for the c-secure CRT code in Section 4. A comparison between the original tracing algorithm and the proposed algorithm by computer simulations is shown in Section 5. In the last section, we conclude our proposal. A preliminary version of this paper appeared in IEICE Technical Report[10].

2. Preliminaries

2.1 Fingerprinting Scheme and Attacks

In the fingerprinting scheme discussed in this paper, an owner of a content embeds a dis-

† Graduate School of Environment and Information Sciences, Yokohama National University

tinct ID, that is an integer ranging from 0 to $n-1$, into the content as a digital watermark so that he obtains slightly different copies to distribute to n users. When a pirated copy is found, the owner is able to trace traitors from the ID extracted from the copy. The embedded data is especially called a fingerprint. The original content is called *cover data.* Various types of digital data can be cover data such as images, videos, audios, texts and software programs. The contents with a fingerprint are called *stego data.*

The biggest threat to the scheme is the attacks that attempt to remove or destroy the fingerprint embedded in the content. The attacks are various but can be divided into two types according to the number of stego data necessary for the attack; one is the attacks with a single stego data and the other is collusion attacks.

The attacks with a single stego data can be considered as a probabilistic alteration over the whole stego data. With only one stego data the attackers do not know where the fingerprint is embedded. Therefore, they simply make an alteration over the whole stego data up to an acceptable level of the content's destruction. For example, compression, scale change, conversion between data formats and cropping against digital watermarkings on images can be considered as attacks with a single stego data. The goal of the fingerprinting scheme is to make it impossible for the attackers to remove or destroy the embedded data without major destruction of the content. So such an alteration that disables traitor tracing will also bring serious destruction to the content. Encoding embedded data using error correcting codes is also proposed to obtain a higher resilience of the scheme against the attacks[4),9)].

Collusion attacks are made by attackers who have obtained two or more stego data. By comparing their stego data, they are able to detect some locations of the embedded data. Then, they scramble only the detected parts so that they do not have to worry about damaging the content. This model of collusion attacks is defined by Boneh and Shaw[1)] as a *marking assumption* (see Subsection 3.3). With collusion attacks, it is more likely for the attackers to succeed in framing innocent users, that is, to generate stego data which are assigned to users who are not in the coalition. Since the quality of contents is not capable of bounding the attacks unlike the case with a single stego data, it is necessary to consider the prevention at the step of the encoding of the embedded data. Therefore, the encoding of embedded data using a collusion-secure fingerprinting code is proposed[1),5),8),11)]. The collusion-secure fingerprinting codes have a tracing algorithm to identify the attackers from the embedded data extracted from the pirated copy.

Now, we discuss a new scenario that the attackers make both attacks: the attack with a single stego data and the collusion attack.

Attackers with two or more stego data first compare their stego data to detect some locations of the embedded data. After scrambling the detected embedded data, they make the second alteration on the whole stego data up to the acceptable level of the content's destruction. The second alteration causes noise on the fingerprint. This means that random errors are added to the embedded data encoded with a collusion-secure fingerprinting code. Therefore, in this scenario, it is necessary to improve the random-error-resilience of collusion-secure fingerprinting codes. In Section 4, we propose a new tracing algorithm of a collusion-secure fingerprinting code that improves the random-error-resilience of the c-secure CRT code.

2.2 Collusion-Secure Fingerprinting Code

In this section, we will describe the properties of collusion-secure fingerprinting codes.

An owner of a content wishes to distribute a distinct fingerprinted copy to n users. He assigns integer $i \in \{0, 1, \ldots, n-1\}$ to user i as a user ID, and embeds it into the content (cover data) to make a stego data to send to user i.

The ID is encoded into a binary data of L bits, where L is a positive integer. We denote the embedded data assigned to user i as $\boldsymbol{w}(i)$. Then code F is called a fingerprinting code.

$$F = \{\boldsymbol{w}(i) \in \{0,1\}^L | 0 \le i < n\}.$$

Let $w(i,j)$ be the jth bit of $\boldsymbol{w}(i)$ then,

$$\boldsymbol{w}(i) = [w(i,0), w(i,1), \ldots, w(i,j), \ldots, w(i,L-1)].$$

Let c be a positive integer smaller than n and T be a coalition of c traitors denoted as $u_0, u_1, \ldots, u_{c-1}$. A set of embedded data (or codewords) assigned to the traitors in coalition T is

$$C = \{\boldsymbol{w}(u_i) | 0 \le i < n\} \subset F.$$

Set C can be considered as a set of IDs assigned to the traitors. Therefore, it is also

called *colluding IDs.* The traitors detect some of the locations of the embedded data by comparing c stego data obtained from coalition T. These locations are called *detected bits.* The traitors are able to replace the detected bits with either 0 or 1 to generate a pirated copy. We denote an altered embedded data generated by C as $\boldsymbol{x}'$. The set of $\boldsymbol{x}'$ that traitors can generate with C is called a feasible set of C[1] and also called a set of descendants of C[7]. We denote the set as $Dec(C)$. Then,

$$\boldsymbol{x}' \in Dec(C) = \prod_{j=0}^{L-1} \{w(u_0, j), w(u_1, j), \ldots, w(u_{c-1}, j)\}.$$

When errors are added to the stego data, error pattern $\boldsymbol{e}$ is added to $\boldsymbol{x}'$. Note that each bit of the error pattern $\boldsymbol{e}$ is added to the corresponding bit of $\boldsymbol{x}'$.

$$\boldsymbol{x}' + \boldsymbol{e} = \boldsymbol{x},$$
$$\boldsymbol{e} = [e(0), e(1), \ldots, e(L-1)],$$
$$\boldsymbol{x} = [x(0), x(1), \ldots, x(L-1)].$$

When random errors of error rate e_r are added, each component of $\boldsymbol{e}$ has an equal probability e_r to have value 1, where $0 \le e_r \le 1$.

Traitor tracing is made by tracing algorithm A, which takes $\boldsymbol{x}$ as an input and outputs the codewords assigned to the traitors. The number of traitors the algorithm identifies is different among the previously proposed codes but the code length tends to be longer when the number of identifiable traitors is larger[5].

The tracing succeeds when algorithm A outputs a subset of C (or C itself when all the traitors are identified). When the output includes a codeword of an innocent user, the tracing fails. Let $e_{fault}(C)$ be the probability of false tracing for all the elements of $Dec(C)$, where $0 \le e_{fault} \le 1$. Then, all the possible coalitions consisting of c or less traitors are considered as follows:

$$for\ all\ C \subset F,\ |C| \le c,$$
$$e_{fault}(C) \le \epsilon.$$

The above condition means that tracing algorithm A is able to identify at least one of the traitors with error ϵ when the coalition consists of c or less traitors. Code F that has such a property is called *c-secure code with error* ϵ or more generally, just c-secure code. In practice, the error ϵ should be small so that the tracing result can be a persuadable evidence of piracy, especially at the accusation of traced traitors.

In the case of $\epsilon = 0$, the code is called *totally c-secure code.* It has been proved by Boneh and Shaw[1] that there is no totally c-secure code when $c \ge n/2$.

3. c-Secure CRT Code

In this section, we explain the construction and the tracing algorithm of the c-secure CRT code[5]. The c-secure CRT code is one of the c-secure codes and has a tracing algorithm that identifies traitors with error ϵ as long as the coalition consists of c or less traitors. The strategies it adopts for reducing the code length are as follows:

1. A distinct integer in $\{0, 1, \ldots, n-1\}$ is assigned to each copy as an ID. An ID is expressed as a set of smaller-sized integers, *residues*, by the Chinese Remainder Theorem (also called Sunzi's Theorem). The c-secure CRT code is a concatenated code. Residues are encoded by their respective inner codes.
2. The n-secure code that has a code length of $O(n)$[11] is adopted as the inner code, because it is the shortest n-secure code for small n.

3.1 Construction

The construction of the c-secure CRT code is as follows:

Modulus Let k, l and m be three positive integers satisfying $\lfloor 2m/c \rfloor = k + l$. Let $p_0, p_1, \ldots, p_{m-1}$ be positive integers which are pair-wise relatively prime satisfying $p_0 < p_1 < \ldots < p_{m-1}$ and $p_0 \times p_1 \times \ldots \times p_{k-1} \ge n$. These integers are called *moduli*. The average of all the moduli is denoted as $\bar{p} = \sum_{i=0}^{m-1} p_i/m$.

Residue Let u be an integer in $\{0, 1, \ldots, n-1\}$. An integer $r_i \in \boldsymbol{Z}_{p_i}$ satisfying $r_i \equiv u \pmod{p_i}$ is called a residue of u modulo p_i, where $i \in \{0, 1, \ldots, m-1\}$. By the Chinese Remainder Theorem, if residues of at least k distinct moduli are given, an integer in $\{0, 1, \ldots, n-1\}$ is determined uniquely, otherwise there is no solution in $\{0, 1, \ldots, n-1\}$ of the simultaneous congruent equations for those moduli.

Inner Code Corresponding to each of the moduli $p_0, p_1, \ldots, p_{m-1}$, m inner codes $\Gamma_0(p_0, t)$, $\Gamma_0(p_1, t), \ldots, \Gamma_0(p_{m-1}, t)$ are respectively prepared. The inner codes are the same

	B_0	B_1	B_2	B_3	B_4
$w_i^{(0)}$	1111	1111	1111	1111	1111
$w_i^{(1)}$	0000	1111	1111	1111	1111
$w_i^{(2)}$	0000	0000	1111	1111	1111
$w_i^{(3)}$	0000	0000	0000	1111	1111
$w_i^{(4)}$	0000	0000	0000	0000	1111
$w_i^{(5)}$	0000	0000	0000	0000	0000

Fig. 1 Example of $\Gamma_0(p_i, t)$ when $p_i = 6$ and $t = 4$.

as the one defined in Ref. 11). Let t be a positive integer and the codewords of $\Gamma_0(p_i, t)$ are given as follows:

$$w_i^{(j)} = \underbrace{000.....00}_{t \times j bits} \ \underbrace{111.....11}_{t \times (p_i - j - 1) bits}, j \in \boldsymbol{Z}_{p_i} \quad (1)$$

Codeword $w_i^{(j)}$ is assigned to user IDs whose residue of modulus p_i is j. Codeword $w_i^{(j)}$ can be divided into $p_i - 1$ blocks of t bits. We call the first group of t bits *block* 0 and denote it as B_0. So the i th block is denoted as B_i. The example of $\Gamma_0(p_i, t)$ in the case of $p_i = 6$ and $t = 4$ is shown in **Fig. 1**.

Outer Code The c-secure CRT code is a concatenated code of the above inner codes. It is denoted as $\Gamma_0(p_0, p_1, \ldots, p_{m-1}; n, t)$. Each user ID in $\{0, 1, \ldots, n-1\}$ is assigned a codeword respectively from m inner codes, $\Gamma_0(p_0, t)$, $\Gamma_0(p_1, t)$,...,$\Gamma_0(p_{m-1}, t)$. A codeword for each user ID is obtained by simply concatenating all the codewords assigned from m inner codes. Then, the codeword for user ID u is as follows:

$$W^{(n)} = w_0^{(r_0)} || w_1^{(r_1)} || \ldots || w_{m-1}^{(r_{m-1})},$$

where $r_i = u \bmod p_i,\ 0 \le i < m$.

The code length L is as follows:

$$L = \sum_{i=0}^{m-1} p_i t = \bar{p} m t.$$

3.2 Tracing Algorithm

Suppose that the owner of the content finds a pirated copy and he extracts the embedded data of L bits. He follows the five steps below to obtain the IDs of the traitors.

Step 1: The owner extracts the embedded data of L bits from the seized pirated copy. The embedded data is denoted as $\boldsymbol{x}$ as it is in Subsection 2.2.

Step 2: As follows, he splits $\boldsymbol{x}$ into m parts in the way that the m parts correspond to each of the m inner codes.

$$\boldsymbol{x} = \underbrace{x_0}_{t(p_0-1)bits} || \underbrace{x_1}_{t(p_1-1)bits} ||.....|| \underbrace{x_{m-1}}_{t(p_{m-1}-1)bits}.$$

The divided part x_i $(i = 0, 1, \ldots, m-1)$ corresponds to inner code $\Gamma_0(p_i, t)$.

Step 3: For each x_i $(i = 0, 1, \ldots, m-1)$, he adopts tracing algorithm A_{in} as below:

Algorithm A_{in}
1: input x_i;
2: for $(min_i = 0; min_i < p_i - 1; min_i + +)$
3: if $(hw_{min}(x_i) > 0)$ break;
4: for $(max_i = p_i - 1; max_i > min_i; max_i - -)$
5: if $(hw_{max-1}(x_i) < t)$ break;
6: output min_i and max_i;

Here, $hw_{min}(x_i)$ and $hw_{max-1}(x_i)$ are the Hamming weight of block B_{min} in input x_i and the Hamming weight of block B_{max-1} in input x_i, respectively. In the tracing, output min_i is expected to indicate the minimum residue and max_i is supposed to indicate the maximum residue of all the residues of modulus p_i that the traitors have been assigned. Therefore, it is called a *residue search*. Furthermore, output min_i and max_i are IDs embedded in some copies in the coalition. We denote the output min_i and max_i as $r_i^{(-)}$,$r_i^{(+)}$, respectively and call them a *residue pair* of inner code $\Gamma_0(p_i, t)$.

Step 4: He counts numbers, $D(u)$ exhaustively for all IDs $u \in \{0, 1, \ldots, n-1\}$, where $D(u)$ is the number of congruent equations which u satisfies with the residue pairs, defined as follows:

$$D(u) = \big|\{i \in \boldsymbol{Z}_m | (u \equiv r_i^{(-)} \pmod{p_i}) \vee (u \equiv r_i^{(+)} \pmod{p_i})\}\big|.$$

$\vee$ is an OR logical operation.

Step 5: We call $D(u)$ a *degree* of the residue pair at u. If the condition $D(u) \ge D_{th}$ is satisfied, the tracing outputs u as a member of the coalition. We call D_{th} a *threshold degree*. The threshold D_{th} is $k + l$, where l is a positive integer as defined in Subsection 3.1. The larger l is, the lower the possibility ϵ of fault tracing becomes. The determination of l is explained in Ref. 5).

3.3 Assumptions

The marking assumption is a precondition for the validity of a c-secure code determined by Boneh and Shaw[1]. On the marking assumption, the attackers can replace only the detected bits with either 0 or 1. The additional assump-

tions for the c-secure CRT code are as follows:

Assumption 1: A coalition is organized randomly.
Assumption 2: A collusion attack generates any embedded data in its feasible set randomly with an equal probability.

Assumption 1 should be reasonable supposing the assignment of IDs are done randomly and kept secret to the users. Assumption 2 is not always reasonable as it is possible for the coalition to decide the values of detected bits statistically, such as a majority decision, etc. However, those are considered to be a stronger version of collusion attacks. In this paper, we exclude those statistical attacks from our consideration and adopt both Assumption 1 and Assumption 2 for further discussions.

3.4 Residue Search

In case of the c-secure CRT code, the blocks detected by a collusion attack on the marking assumption are called *detected blocks*. Tracing algorithm A_{in} for the inner code $\Gamma_0(p_i, t)$ takes input x_i and searches for each end of the sequence of the detected blocks, which indicate the minimum residue $r_i^{(-)}$ and the maximum residue $r_i^{(+)}$ of modulus p_i (see Subsection 3.2 Step 3).

The tracing algorithm checks the Hamming weight of each block from the beginning forward until there is a block B_{min} whose Hamming weight is larger than 0. Then, min_i is outputted as $r_i^{(-)}$. In like wise, the algorithm checks each block from the end backward until there is a block B_{max-1} whose Hamming weight is smaller than t and then outputs max_i as $r_i^{(+)}$.

As long as the attacks are on the marking assumption, the only case that tracing algorithm A_{in} fails the search for the minimum residue is when the end of the detected blocks, which is supposed to indicate the minimum residue, has a Hamming weight of 0 after the attack. Likewise, Algorithm A_{in} fails the search for the maximum residue only when the other end of the detected blocks, which is supposed to indicate the maximum residue, has a Hamming weight of t after being attacked. In both cases mentioned above, the tracing algorithm misses the targeted blocks and outputs wrong residues.

When the attacks are on the marking assumption and also the other two assumptions (Assumption 1 and Assumption 2) defined in Subsection 3.3, the probability that algorithm A_{in} fails to search for the minimum or the maximum residues is $2 \times 1/2^t$. Considering there are m inner codes to search for residues in the c-secure CRT code, the total probability of failed search of the residues is

$$\epsilon_1 = \frac{2}{2^t} m \tag{2}$$

4. Proposal of New Tracing Algorithm

In the previous subsection, we explained that tracing algorithm A_{in} fails in its residue search with the probability ϵ_1 when the collusion attack is made on the assumptions defined in Subsection 3.3.

Now let us consider the new scenario we set in Subsection 2.1, that is, the attackers make both attacks: the attack with a single stego data and the collusion attack. The attackers first make the collusion attacks on the assumptions defined in Subsection 3.3 and generate a new stego data. Then, they make the second alteration on the whole stego data up to the acceptable level of the destruction of the content. The second alteration causes noise on the embedded data. This means that random errors are added to the embedded data.

In this section, we propose a random-error-resilient tracing algorithm of the inner codes of the c-secure CRT code.

4.1 Random-Error-Resilient Tracing Algorithm

Tracing algorithm A_{in} is not designed considering there is a random-error addition on the embedded data simply because the random-error addition is not on the marking assumption. Therefore, it is expectable that the search for the minimum and maximum residues by algorithm A_{in} is distracted by random errors.

The problem is that A_{in} is influenced even by a minor change of Hamming weight of a block (even a change of one bit) and outputs it as the end of the detected blocks, which is actually an undetected block with errors caused by the random-error addition. The matter is how to distinguish detected blocks from undetected blocks with random errors.

We then adopt threshold w_{th} for the distinction of the detected blocks. For the new algorithm we propose, we treat blocks with an alteration of w_{th} or less bits as undetected blocks and blocks with an alteration of more than w_{th} bits as detected blocks. The proposed algorithm A_{th} is shown below:

Algorithm A_{th}
1: input x_i;
2: for $(min_i = 0; min_i < p_i - 1; min_i + +)$
3: if $(hw_{min}(x_i) > w_{th})$ break;
4: for $(max_i = p_i - 1; max_i > min_i; max_i - -)$
5: if $(hw_{max-1}(x_i) < t - w_{th})$ break;
6: output min_i and max_i;

4.2 Efficacy of Proposed Tracing Algorithm

By an adoption of threshold w_{th}, algorithm A_{th} can avoid the influence of all the undetected blocks with w_{th} bits or less errors. By a random-error addition of rate e_r, the probability that one block of length t is added w_{th} bits or less errors is as follows:

$$p_{th} = \binom{t}{1} e_r(1-e_r)^{t-1} + \binom{t}{2} e_r^2(1-e_r)^{t-2} + \ldots + \binom{t}{w_{th}} e_r^{th}(1-e_r)^{t-th} \quad (3)$$

Now, we discuss the disadvantage of the adoption of algorithm A_{th}. Algorithm A_{th} treats only blocks with an alteration of more than w_{th} bits as detected blocks. However, there are detected blocks with an alteration of w_{th} bits or less. Algorithm A_{th} misses these blocks while algorithm A_{in} only misses the detected blocks with no alteration as shown in Subsection 3.4. Therefore the total probability of the failed search of residues increases by ϵ_{th} from ϵ_1 (see Eq. (2)) as follows:

$$\epsilon_{th} = \frac{2m}{2^t} \sum_{i=1}^{w_{th}} \binom{t}{i} \quad (4)$$

We need to consider the efficacy of algorithm A_{th} with the two parameters p_{th} and ϵ_{th} above.

5. Examination

In this section, we examine the efficacy of our proposal. We made a simulation of a collusion attack and a random-error addition against fingerprints encoded by a c-secure CRT code and compared the tracing results by algorithm A_{in} and A_{th}.

We were able to confirm an impressive improvement in random-error-resilience from the tracing result. The procedures of the examination are explained in the following subsection.

5.1 Procedure

The procedure of the examination consists of the following four steps.

Step 1: Generating c-secure CRT Code

We generate a c-secure CRT code with the following parameters:

$$c = 15,\ k = 2,\ p_0 = 100,\ \epsilon = 1 \times 10^{-4},$$
$$l = 5,\ m = 52,\ t = 25,$$
$$n = 1.0 \times 10^4,\ L = 2.77 \times 10^5.$$

Tracing algorithm A_{in} identifies traitors in a coalition consisting of c or less traitors with error ϵ. The moduli are determined by sieving integers starting with the smallest modulus $p_0 = 100$, that is, the other moduli are $p_1 = 101, p_2 = 103, \ldots, p_{51} = 359$. Parameter t is the block length and L is the code length. Parameter c is the maximum number of traitors. Parameter n is the number of user IDs (the number of codewords as well). We eliminate 50 IDs of small integers, that is, $0, 1, \ldots, 49$ because these IDs are falsely traced as a colluding ID with a relatively higher probability. These IDs exist because the distribution of the probability of the residues assigned to each ID is not uniform[5].

Step 2: Collusion Attack Simulation

We randomly form 20 coalitions with q colluding IDs. In each case of $q = 5, 10, 15, 30$, we gain an altered embedded data from each of the 20 coalitions. The collusion attack is on the marking assumption and the other two assumptions defined in Subsection 3.3.

Step 3: Random-Error Addition

We add the random errors with error rate e_r to the altered embedded data. The range of the error rate e_r is from 0.0 to 5.0×10^{-2}.

Step 4: Tracing and Comparison

We input each of the altered embedded data obtained from step 3 into tracing algorithms A_{in} and A_{th}.

From the tracing results, we make a comparison between the two algorithms as follows:

1. CT (Correct Tracing) is the average number of IDs correctly traced by the tracing algorithms.
2. FT (False Tracing) is the average number of IDs falsely traced by the tracing algorithms.
3. RE (Residue Error) is the average number of residues falsely traced by the tracing algorithms for the inner code.
4. P is called *residue error ratio* and calculated by dividing RE by the number of the residue searches that the tracing algorithms make. In this examination, there are 104 residues to search for

Table 1 Determination of threshold w_{th}.

w_{th}	1	2	3
ϵ_{th}	7.7×10^{-5}	1.1×10^{-3}	8.1×10^{-3}
p_{th}	$2.4 \times 10^{-3} \sim 1.1 \times 10^{-1}$	$2.5 \times 10^{-3} \sim 1.17 \times 10^{-1}$	$2.5 \times 10^{-3} \sim 1.17 \times 10^{-1}$

Table 2 Tracing results by algorithm A_{in} ($q = 15$).

e_r	0	1.0×10^{-4}	5.0×10^{-4}	1.0×10^{-3}	5.0×10^{-3}	1.0×10^{-2}	5.0×10^{-2}
CT	9.5	9.05	6.35	4.3	0.25	0	0
FT	0	0	0	0	0	0	0
RE	0	3.5	14.3	25.2	60.0	74.4	91.9
P	0	0.03	0.13	0.24	0.57	0.71	0.88

Table 3 Tracing results by algorithm A_{th} ($q = 15$).

e_r	0	1.0×10^{-4}	5.0×10^{-4}	1.0×10^{-3}	5.0×10^{-3}	1.0×10^{-2}	5.0×10^{-2}
CT	9.5	9.5	9.5	9.4	7.6	4.3	0
FT	0	0	0	0	0	0	0
RE	0	0	0.05	0.4	8.55	24.6	83.4
P	0	0	4.0×10^{-4}	3.8×10^{-3}	8.2×10^{-2}	2.3×10^{-1}	8.0×10^{-1}

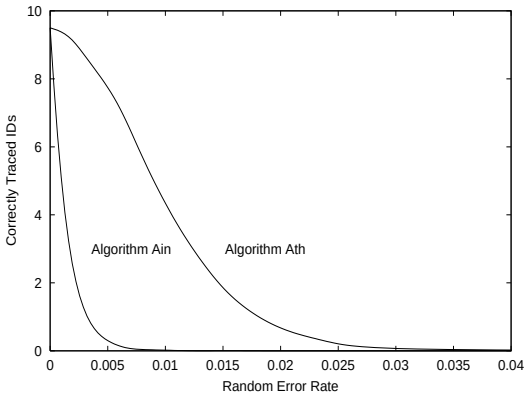

Fig. 2 Comparison of CT between A_{in} and A_{th} ($q = 15$).

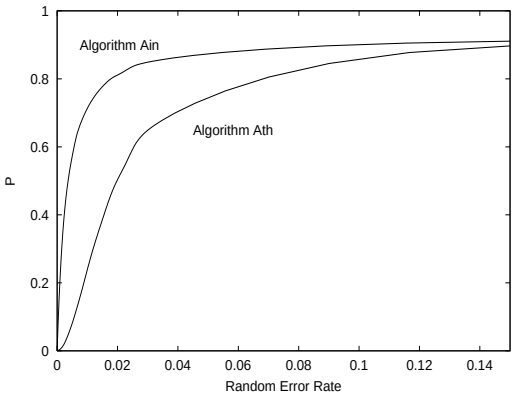

Fig. 3 Comparison of P between A_{in} and A_{th} ($q = 15$).

(52 maximum residues and 52 minimum residues). Therefore, RE is divided by 104 in order to calculate P.

Determination of Threshold w_{th}

We use threshold $w_{th} = 1$ for algorithm A_{th} under the consideration below. Using Eq. (3) and Eq. (4) in Subsection 4.2, ϵ_{th} and p_{th} are calculated in the case of $w_{th} = 1, 2, 3$ as in **Table 1**.

Parameter ϵ_{th} is the increase of the residue error rate caused by the adoption of algorithm A_{th}. Parameter p_{th} is the probability of the existence of an undetected block with errors of w_{th} or less bits, which is avoidable by the adoption of algorithm A_{th}.

Table 1 shows that there is not a big improvement in probability p_{th} by increasing threshold w_{th}. On the other hand, parameter ϵ_{th} increases rapidly as threshold w_{th} is bigger. Therefore, we decide to use $w_{th} = 1$ to receive the most effectiveness at the least sacrifice. Note that the optimal value of threshold w_{th} is derivable in the case that the random error rate e_r can be measured precisely from the altered content by an analysis or other means.

5.2 Results

Table 2 and **Table 3** show the tracing results CT, FT, RE and P, respectively for algorithm A_{in} and A_{th}, in the case of $q = 15$, which is the maximum size of a coalition that the code is designed for.

Figures 2 and **3** show comparisons between the algorithms in CT and P in the case of $q = 15$, respectively.

In Fig. 2, we confirm an impressive improvement in CT brought by the adoption of algorithm A_{th}. For example, the random error rate

Table 4 CT of algorithm A_{in} $(q = 5, 10, 15, 30)$.

e_r	0	1.0×10^{-4}	5.0×10^{-4}	1.0×10^{-3}	5.0×10^{-3}	1.0×10^{-2}	5.0×10^{-2}
$q = 5$	5	5	5	4.8	0.3	0	0
10	9.4	8.95	7.85	5.95	0.2	0.05	0
15	9.5	9.05	6.3	4.3	0.25	0	0
30	2.1	1.9	1.2	0.5	0	0.1	0

Table 5 CT of algorithm A_{th} $(q = 5, 10, 15, 30)$.

e_r	0	1.0×10^{-4}	5.0×10^{-4}	1.0×10^{-3}	5.0×10^{-3}	1.0×10^{-2}	5.0×10^{-2}
$q = 5$	5	5	5	5	4.95	4.75	0
10	9.4	9.4	9.4	9.35	8.75	5.9	0
15	9.5	9.5	9.5	9.4	7.6	4.2	0
30	2.1	2.1	2.1	2.1	1.8	0.9	0

at which the tracing algorithm is still able to trace in average at least one of the 15 traitors is 3.0×10^{-3} for algorithm A_{in}. The error rate for algorithm A_{th} is 1.8×10^{-2}, which is about six times as large as the one for A_{in}.

In Fig. 3, we also confirm an improvement of the residue search of the inner code. Also, we find that CT becomes 0 at around $e_r = 5.0 \times 10^{-3}$ for A_{in} and $e_r = 2.5 \times 10^{-2}$ for A_{th}, which is where P is around 0.6 for both algorithms. This means when the residue search fails over 60%, the c-secure CRT code generated for the examination loses its traceability.

In Table 2 and Table 3, it can be seen that FT is always 0 for both algorithms. It is because the tracing error rate ϵ is too small to be measured by the small number of the simulations made in the examination.

Table 4 and **Table 5** are CT of algorithm A_{in} and A_{th} at $q = 5, 10, 15, 30$. From Table 4 and Table 5, the improvement of the residue search can be seen in each case of $q = 5, 10, 15, 30$. As parameter q approaches to c, algorithm A_{th} becomes more effective. However in the case of $q = 30$, which is double the size of c, CT drops rapidly.

6. Conclusion

We proposed a random-error-resilient tracing algorithm for the c-secure CRT code. We also examined the efficacy of our proposal by computer simulations and confirmed the effectiveness of the proposed tracing algorithm at the error rate range of $10^{-4} \leq e_r \leq 10^{-2}$. This is an explicit example that collusion-secure fingerprinting codes can improve its traceability by only optimizing its tracing algorithm, which can be done after the distributions of stego data.

The reliability of the tracing of fingerprinting codes is ensured only when attacks are done on the assumptions made by the designers of the code. Therefore, as a future work, it may be useful to adopt an analysis of the pirated data to see what kinds of attacks have been made against the fingerprinted copies.

References

1) Boneh, D. and Shaw, J.: Collusion-secure fingerprinting for digital data, *IEEE Trans. Info. Theory*, Vol.44, pp.1897–1905 (1998).
2) Guth, H. and Pfitzmann, B.: Error- and collusion-secure fingerprinting for digital data, *Proc. Third International Workshop, Information Hiding, IH'99*, LNCS, Vol.1768, pp.134–145, Springer-Verlag (2000).
3) Hollmann, H.D.L., van Lint, J.H., Linnartz, J-P. and Tolhuizen, L.M.G.M.: On codes with the identifiable parent property, *Journal of Combinatorial Theory*, A82, pp.121–133 (1998).
4) Iwamura, K., Yamaguchi, K. and Imai, H.: A digital watermark method using public information, *1998 Computer Security Symposium, CSS'98*, pp.33–38 (1998).
5) Muratani, H.: A collusion-secure fingerprinting code reduced by Chinese remaindering and its random-error resilience, *Proc. Fourth Information Hiding Workshop, IHW 2001*, LNCS, Vol.2137, pp.303–315, Springer-Verlag (2001).
6) Stinson, D.R., Trung, Tran van and Wei, R.: Secure frameproof codes, key distribution patterns, group testing algorithms and related structures, *J. Statist. Plann. Inference*, Vol.86, No.2, pp.595–617 (2000).
7) Stinson, D.R. and Wei, R.: Combinatorial properties and constructions of traceability schemes and frameproof codes, *SIAM Journal on Discrete Mathematics*, Vol.11, pp.41–53 (1998).
8) Suzuoki, M., Watanabe, H. and Kasami, T.: A scheme of making collusion-secure watermark, *IEEE-PCM 2000 Conference Proceedings*, pp.328–331 (2000).

9) Yamaguchi, K., Iwamura, K. and Imai, H.: Digital watermarking using error-correction coding for open algorithm, *Proc. 1999 Symposium of Cryptography and Information Security, SCIS'99*, pp.713–718 (1999).
10) Yoshioka, K. and Matsumoto, T.: Random-error-resilient tracing algorithm for collusion-secure fingerprinting code, Technical Report of IEICE, ISEC2001-52, pp.247–254 (2001).
11) Yoshida, J., Iwamura, K. and Imai, H.: A coding method for collusion-secure watermark and less decline, *Proc. 1998 Symposium of Cryptography and Information Security, SCIS'98*, 10.2.A (1998).

(Received November 30, 2001)
(Accepted March 14, 2002)

Katsunari Yoshioka was born in Chiba, Japan on March 28, 1977. He received his Masters degree in Computer Engineering from Yokohama National University in 2002. He is currently a Ph.D. student at the Graduate School of Environment and Information Sciences, Yokohama National University. His research interests cover information hiding and coding theory.

Tsutomu Matsumoto was born in Maebashi, Japan, on October 20, 1958. He received the Dr. Eng. Degree from the University of Tokyo in 1986 and since then his base has been in Yokohama National University where he is enjoying research and teaching in the field of cryptography and information security as a Professor in Graduate School of Environment and Information Sciences. He is a member of Cryptography Research and Evaluation Committee of Japan. He served as the general chair of ASIACRYPT 2000. He is an advisor for Journal of Computer Security and is on the board of International Association for Cryptologic Research. He is a member of IEICE Technical Group on Information Security and of IPSJ Special Interest Group on Computer Security. He received Achievement Award from the IEICE in 1996.