

Improved Tracing Algorithm for Random-Error-Resilient Collusion-Secure Fingerprinting Codes*

CHING-NUNG YANG AND BING-LING LU

*Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, 974 Taiwan*

A randomized c -secure CRT (Chinese Remainder Theorem) fingerprinting code was proposed to avoid the illegal copying problem in digital content distribution systems. The tracing algorithm can detect a collusive member from the fingerprinting code generated by a coalition of c malicious users. However, the collusion attack and random errors increase the tracing error rate. In this paper, we improve the algorithm to achieve a more reliable tracing.

Keywords: fingerprinting code, digital content protection, collusion attack, tracing algorithm, Chinese remainder theorem

1. INTRODUCTION

In digital content distribution systems, the first concern is how to reliably distribute large, rich content to a vast number of heterogeneous receivers via noisy channels. Some error correcting approaches were used to achieve the efficient transmissions, and this is called as a digital fountain [1, 2]. The second is how to avoid illegal copying, *i.e.*, to make such distributions work securely. Therefore, for digital content distribution systems, we care not only the reliability but also the security (the problem of illegal redistributions). Recently, Peer-to-peer network systems, such as BT and their ilk, make it easier to share files, and thus the illegal copying of redistribution on the internet becomes an important issue. If the protected content could be reliably identified as they stream through the cybersphere, then it would be possible to prevent the illegal use.

The fingerprinting code technique is a good solution to address the above piracy problem. We may embed a fingerprinting code of user ID into the digital content. When the content is a pirate copy, the publisher could trace the source of the illegal redistribution by detecting the embedded ID. However, *collusion attack* would compromise the tracing and obtain an incorrect ID. In a *collusion attack*, malicious users collude and compare the embedded content, and then attempt to modify the content to lead to a tracing error.

Nowadays several effective fingerprinting codes [3-7] had been proposed to resist this collusion attack. A c -secure code with ε -error and its corresponding algorithm that can detect at most " c " collusive members with a probability greater than $1 - \varepsilon$ from an illegal fingerprinting code was first proposed in [3]. A lower bound of its code length

Received May 8, 2007; revised April 10, 2008; accepted June 12, 2008.
Communicated by Wen-Guey Tzeng.

* The preliminary version of this paper has been published at P2P Workshop in InfoScale2006, Hong Kong, May, 2006. And this work was supported in part by TWISC@NCKU, NSC under grant No. NSC 97-2219-E-006-009, and the iCAST project under grant No. NSC 97-2745-P-001-001.

satisfying ε was proposed in [8] and further improved by Tardos [9]. The length of Tardos' code will be optimal when satisfying the conditions of Theorem 4 in [9]. However, these theoretical lower bounds of code length were still huge. Therefore, a new such fingerprinting code and evaluation of its "real" code length still deserve studying.

A c -secure code using CRT (Chinese Remainder Theorem) and its variants are described as follows. In [4], the authors used a set of integers which are the residues of CRT to embed the user ID. However, when the random errors are induced, the tracing algorithm results in an incorrect result with a high probability. So it is necessary to distinguish between the random errors and the collusive modifications, and thus a threshold method was proposed in [10]. Subsequently, Watanabe and Kitagawa designed a randomized c -secure CRT code against a new threshold based collusion attack [11]. Their idea is to randomly permute the residues to break the relation between the adjacent IDs and their inner codes. In the Watanabe-Kitagawa algorithm, the backward tracing traces until the value is the same as the residue that the forward tracing did. At this time, a forward tracing error will result in a wrong backward tracing. In this paper, we propose an improved algorithm by processing the forward tracing and backward tracing independently to improve the tracing correctness.

The rest of this paper is organized as follows. In section 2 we describe the randomized c -secure CRT code and the Watanabe-Kitagawa tracing algorithm. In section 3, we design the improved algorithm. Simulation and comparison are given in section 4, and we draw our conclusions in section 5.

2. PRELIMINARIES

2.1 Randomized c -Secure CRT Code

Let n be the number of participants with user ID $u \in \mathbb{Z}_n$ which is expressed by a set of residues of CRT. Residues are randomly permuted and then encoded to the inner codes. The randomized c -Secure CRT code is then constructed by concatenating these inner codes.

Modulus: Let k , k' and l be three positive integers satisfying $\lfloor 2k'/c \rfloor = (k + l)$ where c is the number of collusive members and l is the threshold value chosen as the same condition in [4, 10, 11]. Let $p_0, \dots, p_{k'-1}$ be positive integers which are pairwise relatively prime where $p_0 < \dots < p_{k'-1}$ and $\prod_{i=0}^{k'-1} p_i \geq n$ ($k < k'$). These integers are called moduli and their average value is $\bar{p} = \left(\sum_{i=0}^{k'-1} p_i \right) / k'$.

Residue: Let $u \in \mathbb{Z}_n$ be a user ID. A residue $r_i \in \mathbb{Z}_{p_i}$ is $r_i \equiv u \pmod{p_i}$. If k or more residues are given, the u can be uniquely recovered by CRT.

Random Permutation: Let $P_i(r_i) \in \mathbb{Z}_{p_i}$ be a random permutations of r_i for $0 \leq i \leq k' - 1$. These k' permutation tables are stored secretly in the server for embedding and tracing.

Inner Code: The inner code $w_i^{(p_i(r_i))}$ for the residue $r_i \equiv u \pmod{p_i}$ is constructed by $P_i(r_i)$

t -bit “0” blocks and $p_i - (P_i(r_i)) - 1$ t -bit “1” blocks, shown as follows, where t bit string is a block in the inner code and $t \geq -\log_2(1 - (1 - \varepsilon_i)^{\frac{1}{2k}})$ (one can refer the detail in [11]).

$$w_i^{(P_i(r_i))} = \underbrace{00000 \dots 0}_{t \times (P_i(r_i))} \underbrace{11111 \dots 1}_{t \times (p_i - P_i(r_i) - 1)} .$$

Outer Code: The randomized c -secure CRT code $W^{(u)}$ is a concatenation of k' inner codes shown with the code length $L \sum_{i=0}^{k'-1} p_i t = \bar{p} k' t$.

$$W^{(u)} = w_0^{(P_0(r_0))} \parallel w_1^{(P_1(r_1))} \parallel \dots \parallel w_{k'-1}^{(P_{k'-1}(r_{k'-1}))} .$$

2.2 The Watanabe-Kitagawa Tracing Algorithm

A tracing algorithm is used to obtain the user ID of the malicious redistributors. We first trace two possible boundaries between “0” and “1” from left and right, respectively; subsequently count the number of residues for every ID, and then output the traced user ID when its count number reaches a threshold. The first tracing algorithm [4] only checked the Hamming weight, and any random errors would result in a tracing mistake. The authors in [10] used a threshold w_{th} in t -bit block and ad_{th} adjacent blocks to reduce the effect of random errors but it will be compromised by the threshold based attack. The randomized c -secure CRT code in [11] based on the randomly permutes the residues to overcome this threshold attack.

The Watanabe-Kitagawa Tracing Algorithm

Step 1: Extract the embedded outer code \hat{x} .

Step 2: Splits \hat{x} into k' inner codes:

$$\hat{x} = \underbrace{\hat{x}_0}_{t(p_0-1)\text{bit}} \parallel \underbrace{\hat{x}_1}_{t(p_1-1)\text{bit}} \parallel \dots \parallel \underbrace{\hat{x}_{k'-1}}_{t(p_{k'-1}-1)\text{bit}} .$$

Step 3: For each \hat{x}_i , apply the following procedures:

for ($min = 0$; $min < (p_i - 1)$; $min ++$)
 if ($(H_{min}(\hat{x}_i) > w_{th}) \wedge (H_{min+1}(\hat{x}_i) > w_{th}) \wedge \dots \wedge (H_{min+ad_{th}-1}(\hat{x}_i) > w_{th})$)
 break;
 for ($max = (p_i - 1)$; $max > min$; $max --$)
 if ($(H_{max-1}(\hat{x}_i) < (t - w_{th})) \wedge (H_{max-2}(\hat{x}_i) < (t - w_{th})) \wedge \dots \wedge (H_{max-ad_{th}}(\hat{x}_i) < (t - w_{th}))$)
 break;
 output min and max ;

(Note: $H_{min}(\hat{x}_i)$ is the Hamming weight of the min -th block of \hat{x}_i , where $min \in Z_{p_i-1}$. The threshold value w_{th} is decided according the random error rate. The ad_{th} denotes the number of the successive blocks with Hamming weight greater than w_{th} . The outputs min and max are represented as $r_i^{(-)}$ and $r_i^{(+)}$, respectively, where $r_i^{(-)} \leq r_i^{(+)}$. We call this pair $\{r_i^{(-)}, r_i^{(+)}\}$ a residue pair for the inner code \hat{x}_i .)

Step 4: Count the value $D(u)$ for all $u \in \mathbb{Z}_n$. $D(u)$ is the number satisfying $D(u) = |\{i \in \mathbb{Z}_k \mid (u \equiv P_i^{-1}(r_i^{(-)}) \pmod{p_i}) \vee (u \equiv P_i^{-1}(r_i^{(+)}) \pmod{p_i})\}|$, where P_i^{-1} is the inverse permutation of P_i . $D(u)$ is called as a *degree* of the residue pairs for u .

Step 5: If $D(u) \geq D_{\text{th}}$, outputs u as a collusive member where $D_{\text{th}} = \lfloor 2k'/c \rfloor$ is a threshold degree defined in [11].

Example 1: An inner code x_i with $p_i = 14$, $t = 3$ and the residue $r_i = 7$ is $x_i = (000|000|000|000|000|000|111|111|111|111|111|111)$. Suppose that the tampered inner code \hat{x}_i is $(000|001|010|110|100|000|100|111|001|110|101|111|100)$. The following shows the tracing process.

Let \hat{y}_i be the decoded vector for the inner code \hat{x}_i , using a threshold w_{th} for decoding, *i.e.*, regard the binary t -tuple as “0” for the Hamming weight is at most w_{th} and as “1” otherwise. When choosing $w_{\text{th}} = 1$ and $ad_{\text{th}} = 3$, the inner code \hat{x}_i is first decoded into $\hat{y}_i = (0001000101110)$. By step 3, we obtain $r_i^{(-)} = 9$ and $r_i^{(+)} = 9$. The tracing error occurs since the correct residue is 7. Finally, trace the residues for all inner codes and count $D(u)$. \square

3. IMPROVED TRACING ALGORITHM

There are two major differences between the improved tracing algorithm and the Watanabe-Kitagawa tracing algorithm. The first is that the backward tracing (from the right to the left) will not stop at the value which the forward tracing (from the left to the right) did, but trace until the possible value occurs. The second is that we redefine $D(u)$ in the Watanabe-Kitagawa tracing algorithm.

3.1 Basic Concept

In the Watanabe-Kitagawa tracing algorithm, the ad_{th} successive blocks with Hamming weight greater than w_{th} are used to find the boundary between “0” and “1” in the forward tracing. In the backward tracing, it checks the boundary position with the ad_{th} successive blocks which Hamming weight is less than $(t - w_{\text{th}})$ until the position $r_i^{(-)}$. However, if the combination of random errors and tampering noises occur in a burst type and are located in the 1-block area, the traced position $r_i^{(-)}$ may be at the right side of the correct residue. So, the traced result $r_i^{(+)}$ will be not correct. To overcome this weakness, we loosen the restriction that searching for the ad_{th} successive blocks in the backward tracing stops at the position $r_i^{(-)}$. The following example shows the situation.

Example 2: Consider Example 1, and apply the backward tracing until the condition of successive ad_{th} blocks is met.

In the forward tracing, if there are 3 successive 1-blocks then it stops until $(p_i - 1)$ blocks. In the backward tracing, if there are 3 successive 0-blocks then stops until 0. The forward tracing and backward tracing are shown in Fig. 1. The traced residue pair $\{r_i^{(-)}, r_i^{(+)}\}$ is $\{9, 7\}$. However, the residue pair in Example 1 is $\{9, 9\}$. Note that the correct

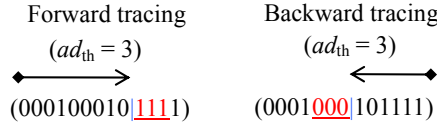


Fig. 1. The forward tracing and backward tracing.

residue “7” is now included in the residue pair $\{9, 7\}$. The reason is that the backward tracing would trace the position through all the inner code. \square

In the new backward tracing, if the residue pair $\{r_i^{(-)}, r_i^{(+)}\}$ is same, we have high confidence on that the inner code is error-free. However, in the Watanabe-Kitagawa algorithm, $D(u)$ counts only once when $r_i^{(-)} = r_i^{(+)}$. So, it is reasonable to give a weighting (count twice) for the most possible residue to improve the tracing correctness. Therefore, $D(u)$ needs to be counted for $(u \equiv P_i^{-1}(r_i^{(-)}) \pmod{p_i})$ and $(u \equiv P_i^{-1}(r_i^{(+)}) \pmod{p_i})$ independently. For example, in the new backward tracing, we find $u \equiv 7 \pmod{14}$ and $u \equiv 9 \pmod{14}$, while we have $\{r_i^{(-)}, r_i^{(+)}\} = \{7, 7\}$ without random error and tampering error. In the latter case, we should fairly count $D(u)$ twice.

3.2 Algorithm

The improved algorithm modifies steps 3 and 4 in the Watanabe-Kitagawa tracing algorithm.

The Improved Tracing Algorithm

Step 3’: For each \hat{x}_i , apply the following procedures:

```

/* forward tracing */
for ( $min = 0$ ;  $min < p_i - 1$ ;  $min ++$ )
if ( $(H_{min}(x_i) > w_{th}) \wedge (H_{min+1}(x_i) > w_{th}) \wedge \dots \wedge (H_{min+ad_{th}-1}(x_i) > w_{th})$ )
break;
/* backward tracing */
for ( $max = p_i - 1$ ;  $max > 0$ ;  $max --$ ) /* “ $max > min$ ” is replaced by “ $max > 0$ ” */
if ( $(H_{max-1}(x_i) < (t - w_{th})) \wedge (H_{max-2}(x_i) < (t - w_{th})) \wedge \dots \wedge (H_{max-ad_{th}}(x_i) < (t - w_{th}))$ )
break;
output  $min$  and  $max$ ;

```

Step 4’-1: Count $D(u) \in \mathbb{Z}_{2k+1}$ for all IDs $u \in \mathbb{Z}_n$ where $D(u)$ is the number of congruent equations which u satisfies $D(u) = |\{i \in \mathbb{Z}_{k'} \mid (u \equiv P_i^{-1}(r_i^{(-)}) \pmod{p_i})\}|$.

Step 4’-2: Count $D(u)$, where $D(u) = |\{i \in \mathbb{Z}_{k'} \mid (u \equiv P_i^{-1}(r_i^{(+)}) \pmod{p_i})\}|$.

Example 3: Consider a fingerprinting code scheme with $c = 2$, $k' = 3$ and $k = 2$. Choose the primes $p_1 = 14$, $p_2 = 17$ and $p_3 = 19$. Then the number of users n is $p_1 \times p_2 = 238$ and the threshold degree $D_{th} = \lfloor 2k'/c \rfloor = 3$. For a user ID = 72, the residues are $r_1 = 2$, $r_2 = 4$ and $r_3 = 15$, respectively. Let the permutations be $P_1(2) = 7$, $P_2(4) = 8$ and $P_3(15) = 6$. Suppose that the tampered \hat{x}_i is first decoded into \hat{y}_i by checking the Hamming weight w_{th} , and the decoded \hat{y}_1, \hat{y}_2 and \hat{y}_3 are $\hat{y}_1 = (0001000101111)$, $\hat{y}_2 = (0010100011111011)$ and $\hat{y}_3 = (000000110110111111)$.

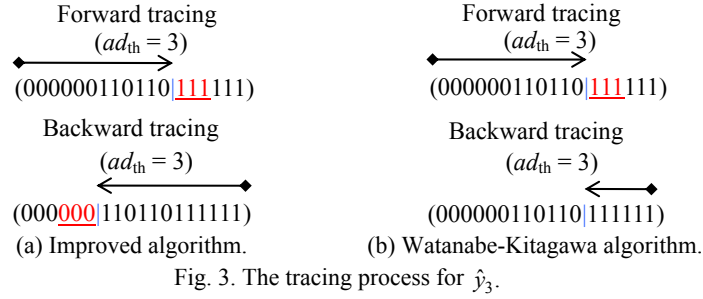
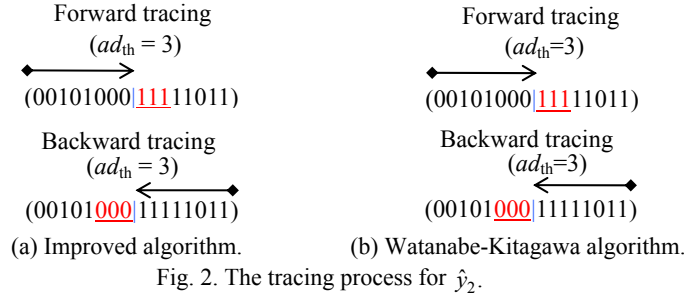


Table 1. The residue pairs.

	The Improved algorithm		The Watanabe-Kitagawa algorithm	
	$\{r_i^{(-)}, r_i^{(+)}\}$	$\{P_i^{-1}(r_i^{(-)}), P_i^{-1}(r_i^{(+)})\}$	$\{r_i^{(-)}, r_i^{(+)}\}$	$\{P_i^{-1}(r_i^{(-)}), P_i^{-1}(r_i^{(+)})\}$
\hat{y}_1	{9, 7}	{11, 2}	{9, 9}	{11, 11}
\hat{y}_2	{8, 8}	{4, 4}	{8, 8}	{4, 4}
\hat{y}_3	{12, 6}	{10, 15}	{12, 12}	{10, 10}

Table 2. The list of $D(u)$.

The Improved algorithm		The Watanabe-Kitagawa algorithm	
$\{P_i^{-1}(r_i^{(-)}), P_i^{-1}(r_i^{(+)})\}$	$D(u)$	$\{P_i^{-1}(r_i^{(-)}), P_i^{-1}(r_i^{(+)})\}$	$D(u)$
{11, 2}	11, 25, 39, 53, 67, 81, ... for 11 2, 16, 30, 44, 58, <u>72</u> , ... for 2	{11, 11}	11, 25, 39, 53, 67, 81, ... for 11
{4, 4}	4, 21, 38, 55, <u>72</u> , 89, ... for 4 4, 21, 38, 55, <u>72</u> , 89, ... for 4	{4, 4}	4, 21, 38, 55, <u>72</u> , 89, ... for 4
{10, 15}	10, 29, 48, 67, 86, 105, ... for 10 15, 34, 53, <u>72</u> , 91, 110, ... for 15	{10, 10}	10, 29, 48, 67, 86, 105, ... for 10

The tracings for \hat{y}_1 by the Watanabe-Kitagawa algorithm and our improved algorithm are shown in Examples 1 and 2, respectively. Figs. 2 and 3 show the tracing processes for \hat{y}_2 and \hat{y}_3 . Suppose the inverse permutations are $P_1^{-1}(9) = 11$, $P_1^{-1}(7) = 2$, $P_2^{-1}(8) = 4$, $P_3^{-1}(12) = 10$ and $P_3^{-1}(6) = 15$. Table 1 lists the residue pairs, and Table 2 summarizes the values $D(u)$. In the improved algorithm, $D(72) = 4$ is greater than the

threshold degree D_{th} . However, the Watanabe-Kitagawa algorithm results in an undetectable tracing for this case, *i.e.* no ID reached the threshold D_{th} . \square

3.3 Discussion

Although we cannot know whether the improved algorithm has the better tracing correctness or not from a couple of examples, we can ensure the correctness of our algorithm. The reason is that we only loosen the searching condition of the backward tracing, and other processes are completely same to the Watanabe-Kitagawa algorithm. In section 4, simulations reveal that our improved algorithm really has better effectiveness than the Watanabe-Kitagawa algorithm. Next, we give a formal analysis to show why the improved algorithm has the better tracing results.

Suppose that a collusion inner code by two colluders A and B is $\overbrace{00\dots 0}^a ** \dots * \overbrace{11\dots 1}^b$. When the random errors and attacked noise are induced, let the tracing results for the improved algorithm be (a', b') where $a' \in [0, p_i - 1]$ and $b' \in [0, p_i - 1]$. There are two cases for the possible values of a residue pair (a', b') : (Case 1) $a' \leq b'$ (Case 2) $a' > b'$. For Case (1) the improved algorithm is same as the Watanabe-Kitagawa algorithm. Table 3 shows all situations of Case (2). Notations, \circ , \times and \square , denote the corrected residue, the erroneous residue and probably correct residue. In Case (2) the improved algorithm performs better than the Watanabe-Kitagawa algorithm for situations (2-2)-(2-5), and has the same performance for situation (2-1). Finally, Case (1) and Case (2) conclude that the improved algorithm has the better tracing results.

Table 3. Detailed classifications for Case (2).

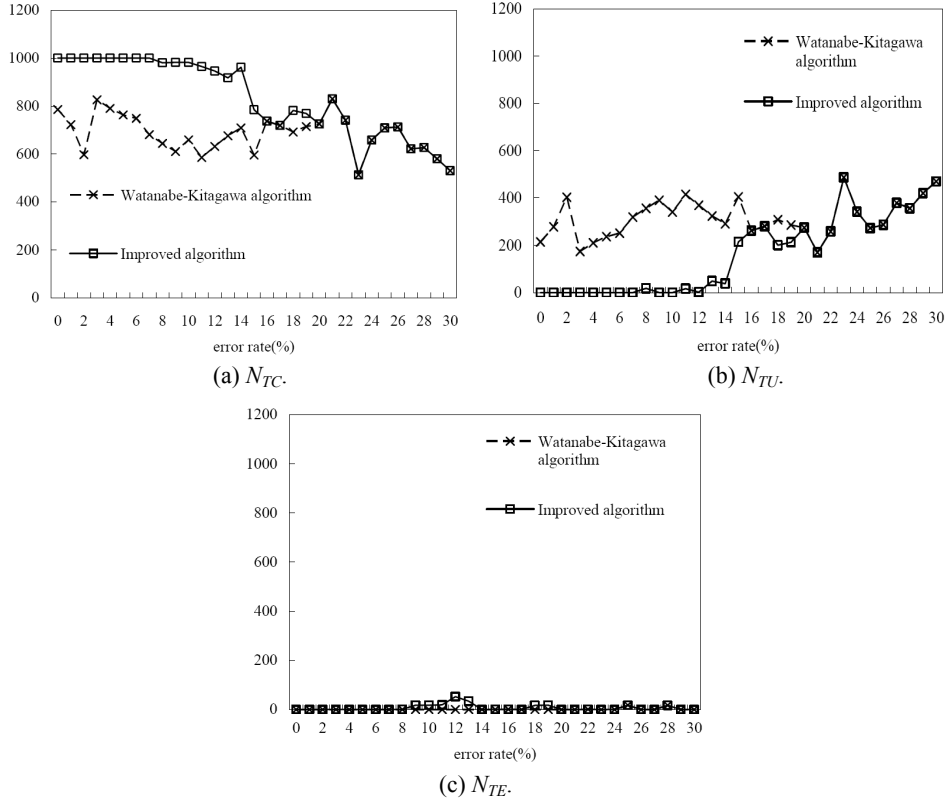
Classification	a' is sectioned in the range $[0, p_i - 1]$	The residue $\{r_i^{(-)}, r_i^{(+)}\}$	
		Improved algorithm	Wat.-Kit. algorithm
(2-1)	$a' \in [0, a)$	$\{\times, \times\}$	$\{\times, \times\}$
(2-2)	$a' = a$	$\{\circ, \times\}$	$\{\times, \times\}$
(2-3)	$a' \in (a, b)$	$\{\times, \square\}$	$\{\times, \times\}$
(2-4)	$a' = b$	$\{\circ, \times\}$	$\{\times, \times\}$
(2-5)	$a' \in (b, p_i - 1]$	$\{\times, \square\}$	$\{\times, \times\}$

4. EXPERIMENTAL RESULTS

There are two major collusion attacks. The first collusion attack is the *uniform selection attack* [3]. The attacker can only change the detected bits by replacing 0 or 1 with equal probability, but he cannot change other places without affecting the fingerprinting code. An uniform selection attack was introduced in [10] that replaces 0 and 1 with the probability p and $1 - p$, respectively, to increase the tracing error. The second is the *threshold based attack* [11]. In this paper, we use the uniform selection attack and the threshold based attack to test our new tracing algorithm. In these simulations, we test 1000 collusion attacks where the parameters of the randomized CRT code are listed in Table 4. Also, N_{TC} represents the number of tracing the correct colluding IDs. N_{TU} represents the number of undetectable tracing, *i.e.*, no ID reaches the threshold degree D_{th} , and N_{TE} is the number of tracing errors. Notice that $(N_{TC} + N_{TU} + N_{TE}) = 1000$. Fig. 4 (a)

Table 4. Parameters of the randomized CRT code.

c	Max number of colluders	15
k', k, l	Parameters (see Modulus)	53, 2, 5
t	Block length (see Inner Code)	25
p_0, p_1	The smallest number	100
n	Number of participant users	1.0×10^4
ad_{th}	Number of successive blocks	3
w_{th}	Hamming weight threshold chosen according random error rate	12

Fig. 4. N_{TC} , N_{TU} and N_{TE} using threshold based collusion attack.

shows that our improved algorithm enhances the tracing correctness when considering the threshold based collusion attack. This improvement is owing to solving the undetectable situations shown in Fig. 4 (b) where the reason can be found in Example 3. At this time the tracing error is almost same as the Watanabe-Kitagawa algorithm (see Fig. 4 (c)). Fig. 4 implies that it is more difficult to fake an inner code to pass the improved tracing algorithm. Fig. 5 shows that both two algorithms have the same resistance to the uniform selection attack. Our improved algorithm resists not only the simple attack (the uniform selection collusion attack) but also the powerful attack (the threshold based attack).

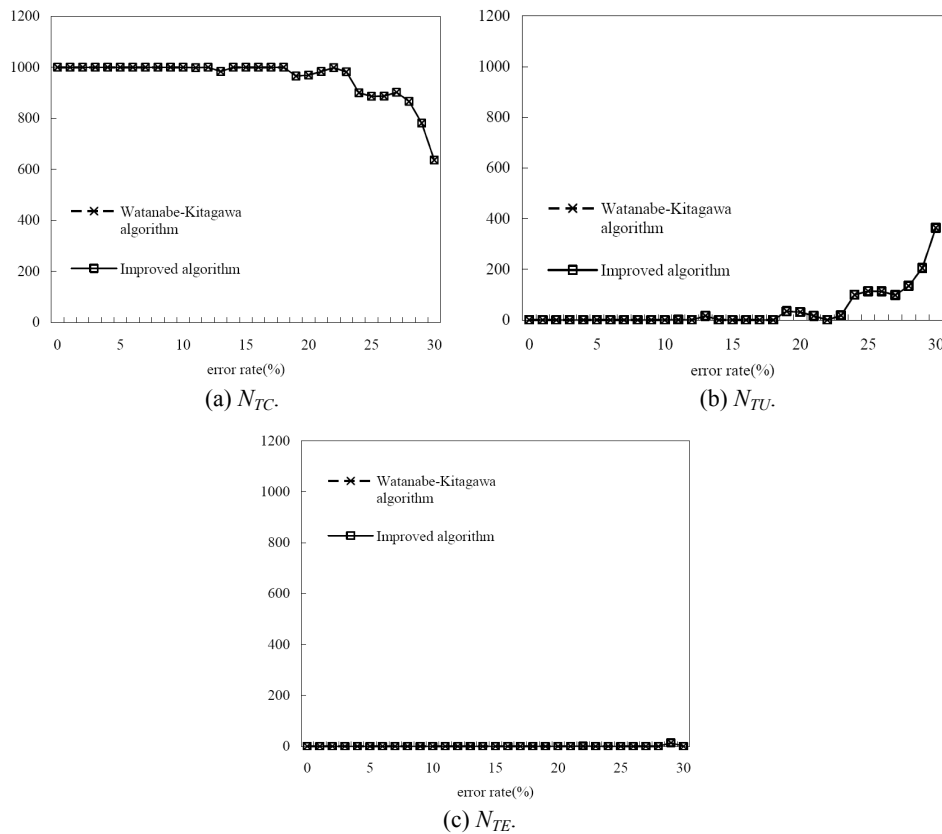


Fig. 5. N_{TC} , N_{TU} and N_{TE} using uniform selection collision attack.

5. CONCLUSIONS

The randomized c -secure CRT code is a good fingerprinting code due to its short code length and can be used to address the illegal copying. By the detail observation of the Watanabe-Kitagawa algorithm, we modified the algorithm in a more reasonable way by applying two strategies: one is the new backward tracing and the other is to redefine $D(u)$. Experimental results show that our improved algorithm traces more correctly. Besides, our improved algorithm makes a small modification and does not increase the complexity.

REFERENCES

1. J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communications*, Vol. 20, 2002, pp. 1528-1540.
2. M. Mitzenmacher, "Digital fountains: A survey and look forward," in *Proceedings of IEEE Information Theory Workshop*, 2004, pp. 271-276.

3. D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," in *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, LNCS 963, 1995, pp. 452-465.
4. H. Muratani, "A collusion-secure fingerprinting code reduced by Chinese remaining and its random-error resilience," in *Proceedings of the 4th International Workshop on Information Hiding*, LNCS 2137, 2001, pp. 303-315.
5. G. Cohen, S. Litsyn, and G. Zemor, "Binary codes for collusion-secure fingerprinting," in *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, LNCS 2288, 2002, pp. 178-185.
6. F. Sebe and J. Domingo-Ferrer, "Short 3-secure fingerprinting codes for copyright protection," in *Proceedings of the 7th Australian Conference on Information Security and Privacy*, LNCS 2384, 2002, pp. 316-327.
7. D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Transactions on Information Theory*, Vol. 44, 1998, pp. 1897-1905.
8. C. Peikert, A. Shelat, and A. Smith, "Lower bounds for collusion-secure fingerprinting," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 472-479.
9. G. Tardos, "Optimal probabilistic fingerprint codes," in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 2003, pp. 116-125.
10. K. Yoshioka and T. Matsumoto, "Random-error resilience of a short collusion-secure code," *IEICE Transactions on Fundamentals*, Vol. E86-A, 2003, pp. 1147-1155.
11. H. Watanabe and T. Kitagawa, "A random-error-resilient collusion-secure fingerprinting code randomized c-secure CRT code," *IEICE Transactions on Fundamentals*, Vol. E86-A, 2003, pp. 2589-2595.



Ching-Nung Yang (楊慶隆) received his B.S. degree in 1983 and the M.S. degree in 1985, both from the Department of Telecommunication Engineering at National Chiao Tung University, Taiwan. He received Ph.D. degree in Electrical Engineering from National Cheng Kung University, Taiwan, in 1997. During 1987-1989 and 1990-1999, he worked at Telecommunication Lab., and Training Institute Kaohsiung Center, Chunghwa Telecom Co., Ltd., Taiwan, respectively. He is presently an associate professor in the Department of Computer Science and Information Engineering at National Dong Hwa University, Taiwan. His research interests include coding theory, information security and cryptography.



Bing-Ling Lu (呂秉霖) received his M.S. degree in 2005 from the Department of Computer Science and Information Engineering at National Dong Hwa University, Taiwan. He is currently a software design engineer in Compal Communications, Inc.