Group-oriented and Collusion Secure Fingerprint for Digital Images

Yongsheng Yu[#], Zhihua Wei
The Green Building Materials and Manufacturing Engineering Research Center of Ministry of Education
Wuhan University of Technology, Wuhan 430070, China
#:yuyosh@hotmail.com, wgaodi@yahoo.com.cn

Xiaosu Chen, Zhiguang Zhang
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan 430074, China
x_s_chen@mail.hust.edu.cn, zgzhang@mail.hust.edu.cn

***Abstract*—Digital fingerprinting is a technique for identifying unauthorized copy and tracing back to its user. The distributor marks each individual copy with a unique fingerprint. The present fingerprinting schemes generally have many difficulties and disadvantages in the case of large-size users. In this paper, we present a new fingerprint scheme which can be used in confidential departments where there are a limited number of users. This scheme is composed of an outer RS (Reed-Solomon) code and an inner code based on BIBD (balanced incomplete block design) code and orthogonal code. We first construct inner code matrix based on BIBD block and orthogonal code. Since the BIBD code just has good performance in resisting linear collusion attacks, we expand the inner code to improve inner code's ability of resisting nonlinear collusion attacks. In reality, the probabilities of participation in collusion in different combinations of users vary. So we group inner codeword into subsets and distribute each subset to a group of users. The code scheme turns into a type of group-oriented fingerprint which is favorable to trace colluders and to avoid involving innocent users. Then we take the inner codeword as the RS code's symbols and make the outer code satisfy the request of resisting collusion attacks. In the end, by concatenating inner and outer code, we get the final fingerprint. The performance of the algorithms is proved and analyzed by theory. Compared with traditional codes for digital fingerprint, under the same collusion size and error probability, our scheme shortens digital fingerprint length significantly, and test experiments confirm the code's ability of resisting collusion attacks.**

***Index Terms*—Digital fingerprint, group-oriented code, collusion attack, BIBD code, RS code, concatenating code**

I. INTRODUCTION

Digital fingerprinting is a technique for a merchant to embed a unique mark into each copy of distributed content. If and when an unauthorized copy of the fingerprinted content is made and distributed, the mark embedded in the copy uniquely identifies the traitor. Such unique marks of the buyers are named fingerprints.

One cost-effective strategy of attacking the digital fingerprints is collusion, where several colluders combine their individual copies to disrupt the underlying fingerprints. By gathering large enough coalition of colluders, it is possible to sufficiently attenuate each of the colluders' fingerprints and produce a new version of the content with no detectable fingerprints. Therefore, collusion-resistant fingerprints are needed to facilitate colluder identification and discourage collusion attempts.

A growing number of techniques have been proposed recently concerning collusion-resistant fingerprint for multimedia. Most of them fall into one of the two categories, according to with or without an explicit discrete coding step. One category is the non-structure fingerprint. A typical example is orthogonal fingerprint, which assigns each user a spread spectrum sequence as a fingerprint, and the sequence is typically orthogonal to those for other users [1]. Non-coded fingerprint is a natural extension of spread spectrum embedding [2] and is easy to implement. Spread spectrum codes that can be used as fingerprints include orthogonal codes, uniformly distributed codes [3], Gaussian random codes, and Welch Bound Equality sequences [4]. Random code for fingerprint is provided in [5]. The combination of the spread spectrum embedding with the error correction coding technology has been studied in [6]. A weakness of non-coded schemes is that the required number of spreading sequences and the computational complexity of detection would increase linearly with the number of users.

The second category, known as structure fingerprint, employs codes and coded modulation to build the fingerprints. Early works on coded fingerprint focused on fingerprint generic data. Boneh and Shaw introduced a two-level construction in code domain to resist up to c colluders with high probability [7]. A number of follow-up works investigate fingerprint codes for generic data with shorter length and faster decoding for digital fingerprint [8]. An optimal probabilistic fingerprint code is shown in [9] and [10] improve its performance. In the past research, identifiable parent property (IPP) code and traceability (TA) code were widely studied [11] [12]. Finite projective geometry and combinatorial tools have also been used to design fingerprint codes [13]. In [14], Trappe *et al.* employ a BIBD code and give an efficient detection method based on tree structure. Many coded

doi:10.4304/jcp.6.2.200-207

schemes have a difficult problem in that the length of the code increases geometrically with the number of users.

The non-structure vs. structure fingerprint is one of the ways to classify all fingerprint methods, yet there are still other ways to group those methods. One influential classifying method is dividing the fingerprint into continuous or discrete fingerprints. A famous continuous fingerprint is provided by Cox *et al.* in [2]. In discrete fingerprint aspect, a fingerprint resisting 2-colluders based on dual binary Hamming codes is provided in [15] and a fingerprint resisting 3-colluders is shown in [16]. In this paper we also consider another classifying way of the normal fingerprint vs. group-oriented fingerprint. In some fingerprint methods, there is an implicit assumption: all random combinations out of the whole users have the same possibilities to take part in collusion attacks. But in reality, the combinations of users who have the same motivation or others relationships have more chances to produce a new illegal version of a multimedia. In [17], Wang and Wu *et al.* put forward a group-oriented fingerprint code which is used for continuous fingerprint.

In some cases, to some degree the number of users is unlimited. In this situation, it is nearly impossible to trace many colluders. Some of the present fingerprints resist a few colluders' collusion attacks, others can resist more colluders but the code length is too long. We consider the protection of confidential images in special departments with a limited number of users. In our work, a fingerprint of short code length can be constructed to resist many colluders' collusion attacks and to trace more even all colluders if possible.

The concatenating code can improve the code performance with constant code length, so we will use concatenating method to generate our fingerprint. In two-level concatenating code, the tracing ability mainly depends on the inner code and the tracing efficiency is chiefly determined by the outer code. Since the BIBD code and orthogonal code have good performance in resisting collusion attacks, we construct our fingerprint based on those two codes. Because the RS code has perfect efficiency in encoding and decoding, we will take RS code satisfying special rules as outer code.

Based on the previous discussion, in this paper, we focus on the structured, discrete and group-oriented digital fingerprint for images. We present a new two-level fingerprint code which is composed of an outer RS code and an inner code based on BIBD code and orthogonal code. We analyze the properties of the code length as well as the collusion security and compare the code length with Boneh's code [7] and Dittmann's code [13] under the same colluder size and error probability. Our fingerprint code can be used to protect the collusion security of videos, too.

The paper is organized as follows. Section 2 describes encoding algorithm. Decoding algorithm is described in Section 3. Section 4 analyzes the performances. Experiment results are given in Section 5. Finally, conclusions are drawn in Section 6.

II. ENCODING ALGORITHM

Boneh and Shaw propose the most robust approach to fingerprinting [7]. It protects against any attacks provided that one assumption holds.

This assumption is named the marking assumption: If a subset of the copies of a protected object agree on some bits of the fingerprint codeword, then these bits are invisible to this subset of users (the colluders), and they cannot modify the bits. Bits on which they disagree are visible and modifiable by the colluders. This marking assumption is also taken in our scheme.

In the following, first we import the traditional BIBD code; then we generate the modifying inner code matrix based on BIBD code and orthogonal code, and expand the inner code matrix. In the end, we derive the outer code and concatenating code.

A. BIBD code

BIBD code is designed originally in the area of combinatorial theory. They are used to arrange elements of a set in which the arrangement possesses certain desired properties in terms of the number of elements per block.

Definition 1: $A(v, k, \lambda)$ balanced incomplete block design (BIBD) is a pair (X,A) , where A is a collection of k-element subsets (blocks) of a v-element set X, and each pair of elements occur together in exactly λ blocks.

$A(v, k, \lambda)$-BIBD has a total of $n=\lambda(v^2-v)/(k^2-k)$ blocks. Corresponding to a block the design is the $v*n$ incidence matrix $M=(m_{ij})$ defined by

$$m_{ij}=\begin{cases}1, \text{if the ith element belongs to the jth block,}\\ 0, \text{otherwise.}\end{cases}$$

An anti-collision code (ACC) is a family of code-vectors in which the bits shared between code-vectors uniquely identify groups of colluding users. An ACC has the property that the composition of any subset of k-1 or fewer code-vectors is unique. This property allows the identification of up to k-1 colluders. A (k-1)-resilient logical *AND* anti-collusion code (AND-ACC) is a code whose composition is an element-wise logical *AND* operation.

Theorem 1: Let (X, A) be a $(v, k+1, 1)$-BIBD and M the corresponding incidence matrix. If the code-vectors are assigned as the bit complement of the columns of M, then the resulting scheme is a k-resilient AND-ACC [14].

The k-resilient AND-ACC from BIBD is called BIBD code. The code length is v, and it can accommodate $n=(v^2-v)/(k^2+k)$ users. When k or fewer users collude, the locations where the BIBD codes agree have a value of one to identify colluders, due to the fact that the set of positions of the sustained 1 is unique.

B. Inner code

In the case of a large number of users, the BIBD code is difficult to construct, and the storage of different composition of BIBD code is very large. So it is unsuitable to built large BIBD block for large number users. Our proposed new fingerprint code includes an

inner code which is constructed with a few BIBD blocks in small size.

The grouping inner code matrix based on BIBD code and orthogonal code is shown in Fig.1. In the figure, B_1 is a BIBD block with size of l_1*c_1; B_2,…, Bp are the same type of BIBD blocks. Apart from those BIBD blocks, there are many zero matrixes with different size. Taking each column from the whole matrix as a vector and taking each vector as an inner code, we can get totally $\sum_{i=1}^{p} c_i$ vectors and the length of a vector is the sum of the number of lines of all BIBD blocks. Putting the vectors which in the same small BIBD block into a group, we can get p groups code. With each BIBD block viewed as a single object, the columns in the inner matrix are totally orthogonal. We can find out: two codes selected at random from different groups are orthogonal with each other; two codes selected at random in the same group maybe not orthogonal based on the property of BIBD code.

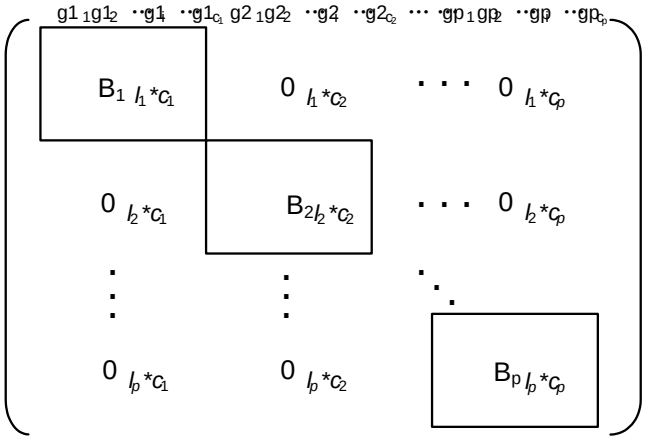


Figure 1. Grouping inner code matrix based on BIBD code

By dividing the users into p subsets, with each subset using a group of codes, and distributing each group of codes to a group of users as their inner code, we get the group-oriented anti-collusion inner code based on BIBD code and orthogonal code.

This type of inner code, being orthogonal in the different groups and un-orthogonal in the same groups, has the following advantages.

First, this inner code is an effective group-oriented code. In the normal code methods, it is assumed that there is the same probability of every user participating in collusion attacks to generate an illegal copy of multimedia. This assumption not only is unfavorable to find out the real colluders, but also involves innocent users. In contrast, the group-oriented code, on the one hand, suits the practical situation better, that is, the users who have close correlations in region or other relationships have more chances to take part in collusion attacks. On the other hand, since the codes in different groups are orthogonal, it is easy to ascertain whether a group of users take part in collusion attacks. That will substantially narrow the scope of suspected users and it can avoid involving innocent users.

Compared with orthogonal code, our code has a far less length under the condition of the same number of users. Assuming that an orthogonal code exists for n users, then one has the code length at least $o(n)$, but in our scheme, the code length is nearly $o(\sqrt{n})$.

Compared with traditional BIBD code, our code has a less computing complex. The construction of BIBD code is very complex, especially in the case of a large number of users. We use small size BIBD block to generate a large inner matrix, which has a low computing complex and can be used for a large number of users.

Using this type of inner code, we should address three issues.

The first is the users' grouping issue. The basic principles about user grouping is that the users who have a high probability to take part in the same collusion attack should be grouped in the same user subset and the users which have a low probability to collusion attack should be grouped in different user subset.

In practical situation, we might quantify the correlations among users. For example, one can quantify the correlation between two users according to how well they know each other, or how far the distance between their locations is, or their work relationship. Upon the quantification of the correlations among users, one can use clustering methods to group them. If the sizes of groups differ significantly, one can merge small groups so that all groups have roughly the same size. Note that it is not allowed to divide large group into small ones when grouping needs adjustment.

The second issue concerns the size of BIBD block in the inner code. The size of BIBD block should be determined by the application requests. Overall, the largest number of colluders which this inner coder can work depends on the size of BIBD block. A very small block size weakens ability of anti-collusion and requires that the inner code length should be increased. On the other hand, if the block is too large, the cost of constructing the BIBD block is huge. The general principle is to employ the BIBD block as small as possible to guarantee the resistance of the possible largest number of users' collusion attacks.

The third issue is the relationships of all BIBD blocks in the inner matrix. Each BIBD block in whole matrix can use the same size block, or one can use different size block to suit the case where sizes of users' groups differs. Using the same BIBD block in the whole inner matrix, we can reduce the computing cost of constructing inner matrix and the same number of inner codeword can be distributed to each users' group.

Since all the other objects values are 0 in the inner code matrix except BIBD blocks, the total number of 0 and the total number of 1 in the matrix differ tremendously. In order to ensure the security and robustness of the fingerprint code, we encrypt the code by using a chaotic series XOR with the inner codes.

The chaotic series is sensitive to initial value, and it is generated from a pseudorandom sequence in practical calculating. We use the Logistic mapping, extensively studied in literatures, to construct real decimals sequences and then convert it to binary number series by section partition in which the particle size depends on the practical application.

Now we provide the construction algorithm of inner code. For the sake of simplicity, all the BIBD blocks in the inner matrix use the same block, i.e., $B_1=B_2=\ldots=B_p$ in Fig. 1. Furthermore, since we focus on the whole anti-collusion concatenating fingerprint, we will not illustrate the details of the construction of BIBD block. Some related content can be found in [14].

Algorithm 1:*InnerCodeEncoding*(*p*,*k*)

Input: A users' group set with *p* groups; largest colluders' number *k*.
Output: A set, *GIC*, storing all grouping inner code.
Function: *BIBDEncoding*(*k*), with input *k* and output the BIBD block.

```
begin
    GIC={};
    B_{r*h}= BIBDEncoding(k);
    Construct the inner matrix which has p BIBD blocks
    with size of r*h and p*(p-1) zero blocks with size of
    r*h, too;
    for each column in G do
        convent into a binary codeword;
        add codeword into group set G_l, each set has h
        codeword;
    generate chaotic series S with length of r*p;
    for each subset in G_l do
        for each codeword g_i in subset do
            g_i'=g_i ⊕ S;
            add g_i' into GIC;
    return GIC;
end
```

Claim 1: (Complexity of InnerCodeEncoding) The time complexity of InnerCodeEncoding (Algorithm 1) is $O(p^2*r*h)$, and the space complexity is $O(p^2*r*h)$, where *p* is the number of users' group and *r***h* is the BIBD block size in inner code matrix.

Proof: In the algorithm, first the BIBD block is generated. Then an inner matrix with a size of (*r***p*)*(*h***p*) is constructed, the time and space complexity is $O(p^2*r*h)$. Convening codeword and doing XOR between codeword and chaotic series involves a time and space complexity of $O(r*h)$. So the total time and space complexity is $O(p^2*r*h)$.

In the methods of collusion attacks, the traditional BIBD code works effectively against logical *AND* collusion attacks, but if the colluders use other collusion attacks methods such as random collusion attacks, BIBD code does not work well. In our method, the inner codes coming from different groups are totally orthogonal, so the final collusion copy will have a few 1 locations no matter which kind of attack is implemented. We can trace some users or groups in collusion with a certain probability in the case of low bit error rate (BER).

In the case of the random attack, we propose a method to improve efficiency. Random collusion attack means that the colluders randomly choose 0 or 1 in those locations where different bits are found. The probability of choosing 0 or 1 is 50%. The improvement is following: First each BIBD code block shall be modified according to the process described in the next paragraph. Then the modified BIBD blocks and zero matrixes are then used to generate the inner code matrix.

Supposing *w* is the codeword of the *k*-resilient BIBD code, $w=(w_1,w_2,...w_v)$, $w_i \in \{0,1\}, (1 \le i \le v)$, the new code is constructed by duplicating each w_i *m* times. We name the chunks of *m* duplicated bits of uniform value "blocks" and use B_i to denote them. Take $B(l,n)$ to express the new code, where *l* is the length of code and *n* is the number of users, then the length of $B(l,n)$ is $l=m*v$. In the following we will prove that the expanding code can resist random collusion attacks.

Claim 2: If the duplicating times *m* satisfies inequality $m \ge \log(v/\varepsilon)/\log 2$, the $B(l,n)$ code is *k*-secure with ε-error.

Proof. When the colluders' fingerprint codes disagree on B_i, they randomly choose bits to create a new B_i. The probability that all *m* bits are 1 is $(1/2)^m$, and this probability equals ε/v, so the w_i will be 0 with a probability of $(1-\varepsilon/v)$. When the colluders' fingerprint codes agree on B_i, the w_i will be 1. So $w=(w_1,w_2,...w_v)$ will be the result of the composition of corresponding BIBD code. The total error probability is $P_e<1-(1-\varepsilon/v)^v \approx \varepsilon$. Because *k*-resilient BIBD code can identify up to *k* colluders, the decoding algorithm will output a colluder with probability at least $1-\varepsilon$.

C. Outer code

We want to design a fingerprint scheme which can resist *k* colluders' collusion attacks. Since the inner code satisfies the request, now we focus on the outer code. Only when both the inner code and the outer code can resist *k* colluders' attack, then the concatenating code is *k*-secure. So the outer code must resist *k* colluder' attack, it means that the distance of outer codeword must satisfy the inequality $D > L/k^2$, where *L* is the length of outer code.

We use one type of error correcting code (ECC) –RS code as outer code of concatenating code [18]. Compared with the other ECC codes, the RS code has more efficient encoding and decoding performance. The encoding method of RS code can use polynomial coding algorithm. We take the Koetter-Vardy soft decision method [19] as the decoding method. The decoding method uses transition probability between send-symbol and receive-symbol in channel to estimate the codes. The input for the method is reliability matrix and the output is a code set including initial inputting codes.

D. Concatenating process

If each group inner code is a subset of codeword of *k*-resilient *B*(*l*,*n*) code, and *w* is *RS*(*L*,*N*,*D*) code whose length is *L*, number of users is *N* ,distance of codeword is *D* and number of symbols is *q*, then as for each codeword $a = \{a1, a2, ..., aL\} \in w$, $ai \in \{1,2,...,n\}, 1 \le i \le L$, let $W_a = w^{(a1)} \| w^{(a2)} \| \cdots \| w^{(aL)}$. || means concatenation of strings. We name the resulting code $\Phi(L, N, l, n)$. Each group of concatenating codes is distributed to the corresponding users, and the total can be used for *n*p* users, where *n* is the number of users in a single users' group and *p* is the number of users' groups.

The code concatenating algorithm is given in the following.

Algorithm 2:*ConcatenatingCodeEncoding*(*p*, *n*, *k*, *m*, *L*)
Input: A users' group set with *p* groups and each group has *n* users; largest colluders' number *k*; inner code duplicate *m* times; outer code length is *L*.
Output: A set, *UIC*, storing all concatenating code.
Function: *RSEncoding*(*I*), with input *I* which is the inner vector and output the RS code *O*.
begin
  UIC={};
  GIC= *InnerCodeEncoding*(*p*,*k*);
  for each inner codeword I_t **do**
    convent into I_t' by duplicate *m* times;
    O_t= *RSEncoding*(I_t');
  for each group p_i **do**
    for each user u_j **do**
      $W_i^j = O_j^1 \| O_j^2 \| ... \| O_j^L$;
      add W_i^j into *UIC*;
  return *GIC*;
end

Claim 3:(Complexity of ConcatenatingCodeEncoding) The time complexity of ConcatenatingCodeEncoding (Algorithm 2) is $O(p^2*r*h)$, and the space complexity is $O(m*p*n*L*r)$, where *p* is the number of users' group, each group has *n* users, *r*h* is the BIBD block size in inner code matrix, inner code duplicates *m* times and outer code length is *L*.

Proof: In the algorithm, a inner code matrix with size of $(r*p)*(h*p)$ is constructed, the time and space complexity is $O(p^2*r*h)$. For RS code encoding and concatenating inner code and outer code, the space complexity is $O(m*p*n*L*r)$. So the total time complexity is $O(p^2*r*h)$ and the space complexity is $O(m*p*n*L*r)$.

III. DECODING ALGORITHM

The decoding process for our code method includes two phases: tracing users' groups within collusion attack and tracing colluders.

Algorithm 3 shows the detail of inner code decoding method, and we put forward the concatenating code's decoding method in algorithm 4.

In the algorithm, first, there is a pretreatment process to perform the collusion codes XOR with the chaotic series. Then we trace the users' groups including colluders. Calculate the number of "1" in the result of logical *AND* between each of the group's character code and the collusion codes extracted from illegal copy of multimedia. If the number is larger than or equal to a certain threshold, the users' group is involved; or else the group is out of the suspected domain.

Algorithm 3:*InnerCodeDecoding*(*C*,*S*,*I*,*m*)
Input: Collusion code *C*, chaotic series *S*, inner code set *I*, inner duplicate times *m*.
Output: A set, *IN*, storing collusion inner code.
Function: *Weight*(*B*), with input binary code *B*, and output the number of "1" in *B*.
begin
  IN={};
  C'=*C* ⊕ *S*;
  divide *C'* to *v* blocks in length of *m*, get block set *B*;
  for each block B_i **do**
    if *Weight*(B_i)=m **then**
      w_i=1;
    else
      w_i=0;
  concatenate w_i, get the collusion codeword w=(w_1, w_2,…,w_v);
  for each character bit in w **do**
    trace collusion codeword (inner code) by BIBD character codeword;
    add collusion codeword into *IN*;
  return *IN*;
end

Algorithm 4:*ConcatenatingCodeDecoding*(*C*,*S*,*I*,*m*)
Input: Collusion code *C*, chaotic series *S*, inner code set *I*, inner matrix *MI* with *r*h* BIBD block ,inner duplicate times *m*, concatenating code set *U*, users' group character code set *P*; users' group number *p*; relating threshold τ, outer code length *L*, the number of largest colluder *k*.
Output: A set, *GI*, storing colluding users' group; set *UI*, storing colluders.
Function: *Distance*(w_1,w_2), with input code w_1,w_2, and output the distance between two codeword.
begin
  GI={}; *UI*={};
  C'=*C* ⊕ *S*;
  for each users' group character code P_i **do**
    if *Weight*(*C'*&P_i) ≥ $m*\tau$ **then**
      add P_i into *GI*;
  IN=*InnerCodeDecoding*(*C*,*S*,*I*,*m*);
  construct inner collusion matrix *M* by *IN*;
  format reliability matrix *R* by *M*;
  decode outer code using Koetter-Vardy soft decision method with *R*; get suspecting code set *U*;
  for each code in *U* **do**
    if *Distance* (U_i,*C*)>(*L*/*k*) **then**
      add U_i into *UI*;
  return *GI*,*IN*;
end

Next split the collusion code into blocks whose length equals m. Use the inner code decoding method to trace the inner codes. Use the result of inner code decoding to construct reliability matrix, use this matrix to detect suspected users by using Koetter-Vardy soft decision method.

In the end, one calculates distance of each suspected code and the collusion code. If the result is equal to or larger than L/k, the user is involved.

Claim 4:(Complexity of ConcatenatingCodeDecoding) The time complexity of ConcatenatingCodeDecoding (Algorithm 4) is $O(p*r)$, and the space complexity is $O(p*v*r)$, where p is the number of user's groups, $r*h$ is the BIBD block size in inner code matrix, and the length of inner codeword is v.

Proof: In the algorithm, first chaotic series XOR with collusion codeword is involved, and the complexity is $O(1)$; then to trace users' group including colluders, one calculates p times; Inner code decoding includes two loops structures and each loop processes v times; to construct a inner collusion matrix, one calculates $p*r$ times; in the end, trace colluder. So the total time complexity is $O(p*r)$ and the space complexity is $O(p*v*r)$.

IV. Performance Analysis

In this section, we analyze our code method performance. First, we prove the code is k-secure code; then we show the method to calculate code length.

A. Code effectiveness

Claim 5: Total users' number is N, distance of outer code is D, inner code matrix is constructed by $r*h$ BIBD blocks, length of inner code is v, the number of outer code's symbol is q, If the duplicating times m satisfy inequality $m \geq \log(2vL/\varepsilon)/\log 2$ and the length of outer code L satisfy: $L=\log_q N+D$, then $\Phi(L,N,l,n)$ code is k-secure with ε-error.

Proof. First, we know the inner code is k-secure with ε-error from claim 2. To observe concatenating code, the distance D between concatenating codes satisfies the inequality $D > L/k^2$. If there are k users taking part in the collusion by logical *AND* attack, then each user must contribute at least L/k bits of his code, which means the distance between colluders' code and the final collusion code satisfies: $Distance(U_i,C)>(L/k)$. Now thinking about an innocent user u arbitrarily, the distance between his fingerprint code and the illegal users' code is no less than D, so the number of same code bits between his code and the final collusion code is less than $k*D=k*(L/k^2)=L/k$, then we have $Distance\ (u,C)<(L/k)$. So the colluders can not incriminate a legal user and we can trace colluders by taking L/k as involving threshold. Note that the duplicating times m satisfies a different inequality compared with claim 2. The reason is that there is only one inner code in claim 2, but in claim 5, there are L inner codes and all codes must satisfy request.

If the colluders use another type of attacking, such as random collusion attack etc., at least one colluder must contribute more bits of his code than L/k, so we can trace him.

B. Code length

We can calculate the length of our code by the following method.

(1). Determine the BIBD block size by the number of largest colluders;

(2). Divide users into groups and get the group number p, construct inner code matrix and get the inner code length v;

(3). Make sure the length of RS code and get $RS(L,N,D)$, where L is the outer code length and N is total users' number, q is the number of outer code's symbol, D is threshold of the outer codes' distance, so $D>L/k^2$ and $N=q^{(L-D)}$, we get $L=\log_q N+D$;

(4). Calculate the inner code duplicate time base on ε: $m = \log(2vL/\varepsilon)/\log 2$;

(5). Calculate code length: $L*m*v*p$.

In [7], Boneh and Shaw introduce a k-secure code with ε-error; in [13], Dittmann come up with another famous anti-collusion code scheme. We compare our code length with their code lengths under the condition of the same collusion attacks.

Tables Ⅰ, Ⅱ show that our proposed coding scheme has shorter length than Boneh-Shaw's code and Dittmann's code under the same colluder size and error probability. (In table 1, each BIBD clock is constructed from BIBD-(16,4,1) and there are two groups of users; in table 2, each BIBD code block is constructed from BIBD-(45,5,1) and all users are divided into two groups, too).

TABLE I. CODE LENGTH COMPARISON UNDER THREE COLLUDERS AND ERROR PROBABILITY 10^{-3}

User number	Our code	Boneh-Shaw's	Dittmann's
100	7,296	454,766	1,010,101
1000	11,552	547,196	1,001,001,001
10000	16,000	640,300	1,000,100,010,001

TABLE II. CODE LENGTH COMPARISON UNDER FOUR COLLUDERS AND ERROR PROBABILITY 10^{-3}

User number	Our code	Boneh-Shaw's	Dittmann's
100	25,200	1,495,008	101,010,101
1000	38,600	1,797,894	1,001,001,001,001
10000	56,700	2,103,303	10,001,000,100,010,001

V. Experiments

In this section, we will test our code scheme's performance of anti-collusion attack by experiment using Matlab software. We take standard gray-scale image "Lena" with size of 512*512 as experiment image. The inner code matrix is constructed by two BIBD blocks whose size is (16,4,1). The number of total users is 100 and all users are divided into two groups, each group has 20 inner codeword. For the purpose of resisting collusion of random attack, each inner codeword duplicates 12 times. Outer code uses RS code with a length of 19, so the final fingerprint code length is 7,296. All initial fingerprints are embedded into the discrete cosine

transform (DCT) [20] domain of images by non-blind watermarking algorithm.

For checking the effectiveness of our code scheme, we took two groups of experiments with collusion attacks in spatial or frequency domain, and in each group, we tested logical *AND* collusion attacks and random collusion attacks. In the spatial domain, we tested two users collusion attack in two situations: two users coming from the same group and coming from different groups. In frequency domain, we tested three users collusion attack with users coming from the same group and coming from different groups, too. For the parameters, we set embedding strength as 0.1 and set the threshold of judging embedded as 0.05.

Experimental Methods:

(1) Collusion attack in spatial domain

Logical *AND* collusion attack: pick up two images which have been embedded two different users' fingerprints, scan every pixel in image, if the values are different in two images' corresponding position, take the logical *AND* value as the new image's pixel value, process each pixel in turn to get a collusion image.

Random collusion attack: pick up two images which have been embedded two different users' fingerprints, scan every pixel in image, if the values are different in two images' corresponding position, take one value in random as the new image's pixel value, process each pixel in turn to get a collusion image.

(2) Collusion attack in frequency domain

Logical *AND* collusion attack: pick up three images which have been embedded three different users' fingerprints, do DCT to each image and get corresponding coefficient; compare three group of coefficient, if they are different, take the coefficients' logical *AND* value as the new image's coefficient, process each coefficient in turn, then do inverse DCT to the resulting coefficients and get a collusion image.

Random collusion attack: pick up three images which have been embedded three different users' fingerprints, do DCT to each image and get the corresponding coefficient; compare three groups of coefficient, if they are different, take one coefficient in random as the new image's coefficient, process each coefficient in turn, then do inverse DCT to the resulting coefficients and get a collusion image.

Experimental results:

Fig. 2 shows the images in experiment, including the image without fingerprint, three users' images (respective user 1, user 2 and user 3 representatives), two collusion images generated by spatial domain collusion attack and two collusion images generated by frequency domain collusion attack.

We do two groups of experiments, fist group include 100 collusion attacks in spatial domain coming from two users which are selected in random, and the second group include 100 collusion attacks in frequency domain coming from three users which are selected in random, too.

In 82 experiments which involves logical *AND* collusion， we can trace every colluder in the logical *AND* collusion attacks in both spatial domain (in 51 experiments) and frequency domain (in the other 31 experiments).

In experiments of random collusion attacks in spatial domain, we can trace two colluders when two users coming from different user's groups; we can trace two colluders in 28 experiments out of all 49 experiments, and trace one colluder in the other 21 experiments.

Figure 2. (a) Image without fingerprint, (b)(c)(d) image of user 1,2,3, (e) image coming from two users' logical *AND* collusion attack in spatial domain, (f) image coming from two users' random collusion attack in spatial domain, (g) image coming from three users' logical *AND* collusion attack in frequency domain, (h) image coming from three users' random collusion attack in frequency domain.

In experiments of random collusion attack in frequency domain, there are one user in a users' group and two users in another group in 69 experiments. In this

situation, the user who is along in a group can be traced in every experiments, and the other two users who are in the same group both can be traced in 42 experiments and only one can be traced in 27 experiments. In 31 experiments, three users are coming from the same group. The result is that we can trace three colluders in 15 experiments, trace two colluders in 11 experiments and trace one in the other 5 experiments.

Overall, we can trace all colluders in logical *AND* collusion attack; in most cases we can trace several colluders in other method of collusion attack; and in few cases we just can trace only one colluder.

VI. CONCLUSION

In this paper, we study the problem of confidential multimedia protection in departments with a limited number of users by using special digital fingerprint resisting collusion attacks.

First we construct the inner matrix based on BIBD code and orthogonal code. Then, in view that the BIBD code just has good performance in resisting linear collusion attacks, we expand the inner code by duplicate each bit a few times to improve the ability of resisting nonlinear collusion attacks. Since the probability of different users' participation in collusion attack varies, we group the users and divide the inner codeword set into subgroups based on its location in inner matrix. Then we get a group-oriented inner code scheme. We take special RS code satisfying certain rules to resist collusion attacks, and concatenate two types' codes to generate the final fingerprint for users. We also propose efficient algorithms to construct the fingerprint.

In the paper, we show the details of decoding process which includes two stages: to identify the collusion users' group and to trace the final colluders. We provide the corresponding algorithms. Compared with the other two famous code schemes in the same collusion situation, our code has less code length. In the end, experiments confirm that our code can resist collusion attacks.

REFERENCES

[1] Z.J. Wang, M. Wu, H. Zhao, W. Trappe, and K.J.R. Liu,"Anti-Collusion Forensics of Multimedia Fingerprinting Using Orthogonal Modulation", IEEE Trans. on Image Proc., 14(6), pp.804-821, 2005.

[2] I. Cox, J. Kilian, F. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", IEEETrans. on Image Processing, 6(12), pp.1673-1687, 1997.

[3] J. Kilian, T. Leighton, L. Matheson, T. Shamoon, R. Tarjan, and F. Zane, "Resistance of digital watermarks to collusive attacks," in Proc. IEEE Int. Symp. Inform. Theory, Aug. 1998, pp. 271.

[4] Z. Li and W. Trappe. "Collusion-resistant fingerprints from WBE sequence sets", In Proc. IEEE ICC 2005, Seoul, Korea, May, pp. 16–20 2005.

[5] F. Zane, "Efficient watermark detection and collusion security," FC 2000, Lecture Notes Comput. Sci 1962, pp. 21–32, 2001.

[6] J. Lofvenberg, N. Wiberg. Random Codes for Digital Fingerprinting. Technique Report, LiTH-ISY-R-2059, Department of Electrical Engineering, Linkoping University, 2000, http://www. it.isy.liu.se/jacob/texter/RandCodes

[7] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data", IEEE Trans. Inform. Theory, vol. 44, pp. 1897–1905, Sept. 1998.

[8] A. Barg, G.R. Blakley and G. Kabatiansky "Digital fingerprinting codes: Problem statements, constructions, identification of traitors", IEEE Trans. Inform. Theory, 49(4), pp. 852-865, April 2003.

[9] G. Tardos, "Optimal probabilistic fingerprint codes", Proceedings of the 35th Annual ACM Symposium on Theory of Computing, pp. 116-125,2003.

[10] K. Nuida1, S. Fujitsu, M. Hagiwara, et al. An Improvement of Tardos's Collusion-Secure Fingerprinting Codes with Very Short Lengths. In: Proceedings of AAECC 2007, Springer-Verlag, 2007. pp. 80~89

[11] D. To, R. Safavi-Naini and Y.Wang, "A 2-secure code with efficient tracing algorithm", Progress in Cryptology-INDOCRYPT'02, Lecture Notes in Computer Science, Vol. 2551, Springer-Verlag, pp. 149-162, 2002.

[12] J.N. Staddon, D. R. Stinson, and R. Wei, "Combinatorial Properties of Frameproof and Traceability Codes", IEEE Trans. on Information Theory, vol. 47, no. 3, pp. 1042-1049, March 2001.

[13] J. Dittmann, P. Schmitt, E. Saar, J. Schwenk, and J. Ueberberg, "Combining digital watermarks and collusion secure fingerprints for digital images," SPIE J. Electron. Imaging, vol. 9, no. 4, pp. 456–467, 2000.

[14] W. Trappe, M.Wu, Z. J.Wang and K. J. R. Liu. "Anti-collusion fingerprinting for multimedia". IEEE Trans.Signal Proc., 51(4). pp. 1069–1087, Apr. 2003.

[15] J. Domingo-Ferrer, J. Herrera-Joancomartí. A Simple Collusion-secure Fingerprinting Schemes for Images. In: Proceedings of the International Symposium on Information Technology: Coding and Computing (ITCC 2000), IEEE Press, 2000. pp. 128~132

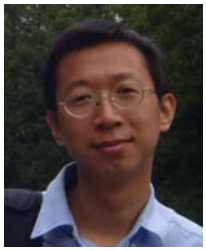
[16] F. Sebe, J. Domingo-Ferrer. Collusion-Secure and Cost-Effective Detection of Unlawful Multimedia Redistribution. IEEE Transactions on System, Man, and Cybernetics-Part C, 2003, 33. pp. 382~389

[17] Z. JaneWang，Min Wu，W. Trappe, and K. J. R. Liu. "Group-Oriented Fingerprinting forMultimedia Forensics". EURASIP Journal on Applied Signal Processing, 2004:14. pp. 2153–2173, 2004.

[18] R. Koetter, A. Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. IEEE Transactions on Information Theory, 2003, 49. pp. 2809~2825

[19] A. Vardy. Bit-level Soft-Decision Decoding of Reed-Solomon Codes. IEEE Transactions on Communications, 1991, 39(3). pp. 440~444

[20] A. H. Taherinia, M. Jamzad. A Robust Image Watermarking Using Two Level DCT and Wavelet Packets Denoising. In: Proceedings of 2009 International Conference on Availability, Reliability and Security, Fukuoka, 2009. pp. 150~157



Yongsheng Yu was born in Hongan county, Hubei province, China, July 15, 1978. He received his **B.S.** degree in Computer Science in Wuhan University of Hydraulic and Electrical Engineering in 2001. He obtained his **M.S.** and **Ph.D.** degrees in Computer Science from Huazhong University of Science and Technology in 2005 and 2010, respectively. He is currently a postdoctoral researcher in The Green Building Materials and Manufacturing Engineering Research Center of Ministry of Education at Wuhan University of Technology. His research interests include computer security, image processing and computer simulation.