

Binary Forensic Code for Multimedia Signals: Resisting Minority Collusion Attacks

W. Sabrina Lin^a, Shan He^{*b}, Jeffrey Bloom^b

^aECE Department, University of Maryland, College Park, MD 20742, U.S.A;

^bThomson Corporate Research, 2 Independence Way, Princeton, NJ 08540, U.S.A

ABSTRACT

Digital forensic marking is a technology to discourage unauthorized redistribution of multimedia signals by embedding a unique mark into each user's copy of the content. A powerful class of attacks on forensic marking is the collusion attack by a group of users. Recently, a new collusion attack, called the minority attack, has been proposed against forensic marking schemes with correlation-based detectors. Although this attack is not very effective on Gaussian-based forensic marking, it is quite powerful on removing the traces of users when the forensic marking is binary. In this paper, we first study the performance of an ECC-based binary forensic code under the minority attack and we model the additional processing, such as compression, applied on colluded copy as a binary symmetric channel. We confirm that the system can be defeated by a minority attack from only 3 colluders. To resist the minority attack, we propose a row-permuted binary orthogonal code to serve as the inner code for ECC-based forensic code, coupled with an adaptive detector. Experimental results show that the proposed scheme has a significantly improved resistance to a minority attack.

Keywords: Collusion Resistance, Multimedia Forensic Marking, Binary Forensic Code, Minority Collusion Attack.

1. INTRODUCTION

Digital forensic marking is a technique for identifying users whose copy of a multimedia work is used for unintended purposes, such as unauthorized redistribution. Forensic codes are typically embedded into the content using watermarking techniques that are designed to be robust to a variety of attacks. A cost-effective attack against such digital forensic codes is collusion, in which several differently marked copies of the same content are combined to disrupt the underlying watermarks. While collusion attacks are effective against general forensic codes, they become a special challenge when the protected data is multimedia and the colluders can apply post-processing after collusion to further degrade the recovery. For instance, the result of a collusion attack may be compressed to reduce the data size for efficient redistribution. Therefore, it is important to design collusion-resistant forensic codes that are also robust to channel error.

A number of collusion attacks have been studied in the forensic marking literature. The most representative are averaging collusion and interleaving collusion [1]. In addition to its inherent effectiveness, averaging collusion can also be used to model many non-linear collusion attacks under the assumption of Gaussian marking [2]. Interleaving attacks are implied in most of the forensic code works [3]-[5]. Recently, a collusion attack, called the minority attack, has been proposed against forensic marking with correlation-based detectors [6]. Although this attack is not very effective on Gaussian-based forensic marking, such as [1], it is quite powerful on removing the traces of users when the forensic marking is binary.

In this paper, we first study the performance of an ECC-based binary forensic code under the minority attack. We use a binary symmetric channel to model degradations due to additional processing applied to the colluded multimedia signal. We then modify the ECC-based code by introducing a row-permuted binary orthogonal code as the inner code. Coupled with this inner code design, we develop an adaptive detector which can be switched between hard detection and soft detection based on the detection statistics. Preliminary results show that the proposed scheme has a significantly improved resistance to minority attacks.

* shan.he@thomson.net; phone 1 609 987-7741; fax 1 609 987-7363; thomson.net

2. OVERVIEW OF ECC-BASED FORENSIC MARKING CODES

2.1 Code Construction

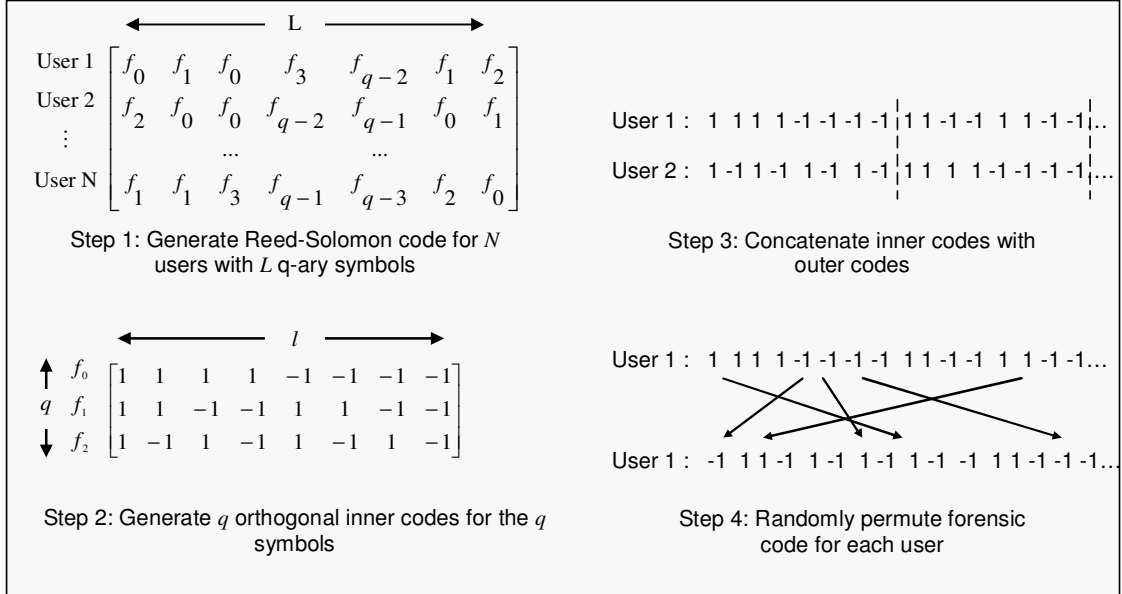


Figure 1. Illustration and example of an ECC-based forensic code construction. In Step 1, N Reed-Solomon codewords of length L are generated, each composed of q -ary symbols. This is known as the outer code. In Step 3, each symbol is replaced with a binary inner code word of length l , where all of the inner code words are orthogonal. Finally, the user codewords are randomly permuted yielding N , binary user codewords that are each length $l \times L$.

Figure 1 shows the process of generating ECC forensic codes. The first step is to generate an ECC outer code, which is constructed as a Reed-Solomon code in this paper, for N users with L q -ary symbols, $\{f_0, f_1, \dots, f_{q-1}\}$. Each of the q symbols in this code is then replaced with one of q orthogonal binary inner codewords taking values of ± 1 . Finally, as discussed in [1], we randomly permute each forensic codeword to prevent the code structure from being exploited by attackers. The overall code length is $l \times L$ bits and the total number of supported users is $N = q^l$, where l is the dimension of the outer Reed-Solomon code.

The orthogonal inner code studied here is the Kronecker product of an exponential orthogonal code and the Hadamard matrix explored in our earlier work [7]. This inner code has been shown to have good resistance to the majority attack and to the interleaving attack¹. Here the Kronecker product of X and Y , denoted by $X \otimes Y$, is defined as follows:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad X \otimes Y = \begin{bmatrix} Y & Y \\ Y & -Y \end{bmatrix}.$$

The Hadamard matrix H_{q_c} with order q_c is a $q_c \times q_c$ orthogonal matrix which exists when $q_c = 2^m$ and m is an integer. It can be generated recursively by

¹ The Majority attack takes the majority of the bits from colluders at each bit position. The Interleaving attack randomly picks the bit from one of the colluders at each bit position [7].

$$H_{q_c} = \begin{bmatrix} H_{q_c/2} & H_{q_c/2} \\ H_{q_c/2} & -H_{q_c/2} \end{bmatrix}, \text{ with } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Exponential orthogonal codes are designed to preserve the colluders' information as much as possible. The columns of the exponential orthogonal code matrix consist of all 2^q possible combinations of 1 and -1, one column corresponding to the bits from q codewords at a given bit position. Thus, the code length is 2^q . The exponential orthogonal code is constructed as follows: for the i^{th} codeword, f_{i-1} , the first 2^{q-i} bits are 1 and the next 2^{q-i} bits are -1. Then we repeat the same code for 2^{i-1} times, ending up with 2^q bits. An example exponential code matrix for $q = 3$ is shown in step 2 of Figure 1.

2.2 Detection

A "soft" maximum correlation detector is employed in this ECC forensic code scheme. Soft detectors are widely used for forensic codes as they tend to perform better than hard detectors under majority and interleaving attacks [1].

Let y be the forensic code extracted from the colluded copy after an inverse permutation process. This permutation is the inverse of that done in Step 4 of Figure 1. The maximum correlation detector calculates the correlations between each symbol in the extracted vector and a normalized version of each of the codewords in the code book. This yields a $q \times L$ array of correlations where each correlation is between an extracted segment of length l and one of the q normalized inner codes.

Once the correlation array is built, it is used to determine which copy (user) is most likely to be a contributor to the captured copy as follows. For a given user, i , calculate the average correlation for the symbols in the outer code assigned to user i . Then identify the user with the highest such average correlation. This can be stated more formally below.

Let x_i be the forensic code assigned to user i and U be the set of all users. The detection statistic for user i is then T_i , calculated as follows:

$$T_i = \sum_{j=1}^L T_i^{(j)} / L, \text{ with } T_i^{(j)} = \frac{\langle y^{(j)}, x_i^{(j)} \rangle}{\|x_i^{(j)}\|}, \quad (1)$$

where $x_i^{(j)}$ and $y^{(j)}$ are the codewords for the j^{th} symbol of x_i and y , respectively. User i is identified as a colluder if T_i is the maximum among the detection statistics of all users, i.e. $T_i \geq T_k, \forall k \in U$.

3. THE MINORITY ATTACK

3.1 Interleaving Attacks

Interleaving attacks assume that the colluders have access to the bit locations of their marked copies and that the bit positions do not vary from one copy to the next. If the content at one bit location is the same in all copies, then the colluders may assume that the bit embedded at that position is the same in all copies; although they will not know the value of that bit. When the content at a bit location is different from one copy to the next, the colluders will be able to select one from the set of values they have. There are a number of interleaving attacks; the most commonly discussed are the random interleaving attack, usually referred to as interleaving attack for simplicity, where the colluders pick at random one of the colluders' bit value, and the majority attack, where the colluders pick the value that appears most often. In a recent paper, Schaathun presents the Minority attack, where the colluders pick the value that appears least often [6].

3.2 Minority Attack on Forensic Codes

The minority attack by c colluders for ± 1 binary forensic code is formulated as follows [6]: at bit position j in all c copies from the colluders, denote the number of -1's as $n_{-j}^{(c)}$, and the number of 1's as $n_{+j}^{(c)}$. Then the minority colluded code at position j is

$$y_j = \begin{cases} -1, & \text{if } n_{-1}^{(j)} \leq n_1^{(j)} \\ 1 & \text{otherwise} \end{cases}$$

The rationale behind this attack may seem intuitive. The correlation based detector collects each attacker's trace from every code bit of the colluded copy. By choosing the bit at each position that appears least often, the colluders hope to reduce the correlation, and thus increase the probability of detection error.

In order to confirm the asserted vulnerability of the ECC forensic code to this attack, we've implemented the code introduced in Section 2 as follows. The outer code is a RS code with $q=32$ symbols and dimension $t = 4$, thus supporting q^t users (just over a million) with length $L=31$. The inner code is the Kronecker product of a 4×4 Hadamard matrix and an exponential orthogonal code with 8 codewords. The resulting inner code has length $l=2^{10}$.

We performed the minority attack followed by a binary symmetric channel (BSC) noise source to introduce a bit error rate of δ and studied the performance of the code under channels with various bit error rates when there are 3, 4 or 5 colluders. (Note that the 2 colluder case is interesting, but not applicable for study of the minority attack.) For each case (c randomly selected colluders and $\text{BER} = \delta$), we performed 200 trials.

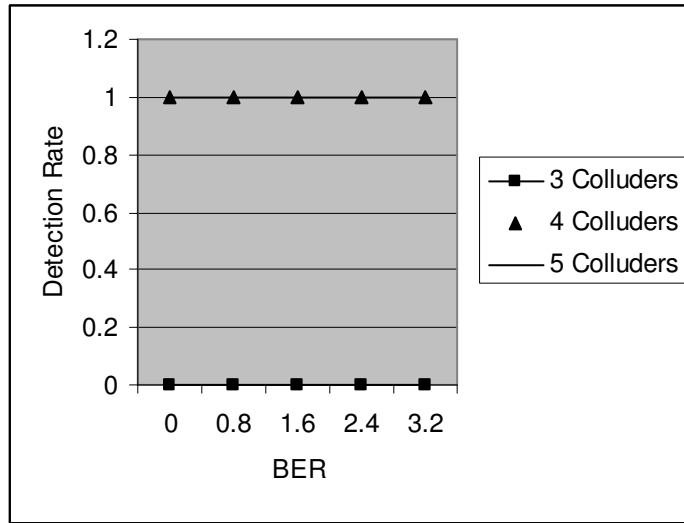


Figure 2. Detection Rate of the ECC code against minority attack by 3, 4, or 5 colluders with additional BER of 8%, 16%, 24%, and 32%.

Figure 2 shows some results from this study. The detector identifies the user, from the q^t users, most likely to be a contributor to the colluded copy. When there are 4 or 5 colluders and $\text{BER}=0$, the identified user is always one of the colluders. For these tests, we say that the detection rate is 100%. This performance holds for BER upto and including 32%. However, the detection rate falls to 0 when there are exactly 3 colluders. This is true even without the addition of BSC noise ($\text{BER}=0$ case). In the following section, we analyze this special case to understand why the minority attack by 3 colluders is so powerful against the ECC-based binary forensic code.

3.3 Attack Analysis

A close analysis of the structure of the binary orthogonal code leads to the following theorem.

Theorem 1: Given any three binary orthogonal codewords, x_1 , x_2 , and x_3 , which take values of ± 1 , the minority-colluded codeword y is orthogonal to x_1 , x_2 , and x_3 , i.e. $\langle y, x_1 \rangle = \langle y, x_2 \rangle = \langle y, x_3 \rangle = 0$.

Proof: Let l be the length of the codewords and $x_1(i)$, $x_2(i)$, $x_3(i)$, and $y(i)$ be the i^{th} bit of x_1 , x_2 , x_3 , and y , respectively. Without loss of generality, we examine $x_1(i)$. There are three possibilities for the value of $x_1(i)$: it is the minority among $x_1(i)$, $x_2(i)$, and $x_3(i)$; it is the majority among the three bits; it has the same value as $x_2(i)$ and $x_3(i)$.

If $x_1(i)$ is the minority, then $y(i) = x_1(i)$, and thus $y(i) \times x_1(i) = 1$. Furthermore, since $x_1(i)$ is the minority, $x_2(i)$ must be equal to $x_3(i)$. Therefore, $x_2(i) \times x_3(i) = 1 = y(i) \times x_1(i)$.

Similarly, if $x_1(i)$ is the majority, then $y(i) = -x_1(i)$, and $x_2(i) = -x_3(i)$. We have $x_2(i) \times x_3(i) = y(i) \times x_1(i) = -1$.

Finally, if $x_1(i) = x_2(i) = x_3(i)$, apparently $y(i) = x_1(i) = x_2(i) = x_3(i)$, and $x_2(i) \times x_3(i) = y(i) \times x_1(i) = 1$.

Hence, $x_2(i) \times x_3(i) = y(i) \times x_1(i)$ in all three cases. We can extend this to say that $x_2(i) \times x_3(i) = y(i) \times x_1(i)$ for all $1 \leq i \leq l$. Now, consider the inner product of y and x_1 : $\langle y, x_1 \rangle = \sum_{i=1}^l y(i) \times x_1(i) = \sum_{i=1}^l x_2(i) \times x_3(i) = \langle x_2, x_3 \rangle = 0$. The last equation holds because of the orthogonality of x_2 and x_3 .

The same proof can be applied to get $\langle y, x_2 \rangle = \langle y, x_3 \rangle = 0$. □

Based on Theorem 1, the code resulting from a minority attack on any binary orthogonal forensic code using a correlation-based detector is always orthogonal to the colluders' forensic codes when the number of colluders is 3. This can occur in the inner code described in Section 2.1. Consider the case of 3 colluders who each have different symbols at symbol location i in the outer code. Associated with each symbol is a different inner code selected from an orthogonal binary codebook. The result of the minority attack at this symbol location will be orthogonal to each of the three contributing symbol codes, but not necessarily to the other $q-3$ symbol codes. This will have the effect of decreasing the detection statistic of the colluders' and increasing the detection statistic of non-colluders.

4. RESISTING THE MINORITY ATTACK

The problem exposed in the previous section is that the result of a minority attack on the inner code is orthogonal to the codes of the three contributing symbols. In order to resist this attack, we need to develop an inner code that does not have this property. The approach is to create an inner code that is a concatenation of a number of subcodes. The collusion may be orthogonal to one subcode in an inner code, but not the entire inner code word.

4.1 Row-permuted Inner Binary Code Design

In this section, we propose a modification to the inner binary code design in order to resist the minority attack. The previously described inner code was based on the Kronecker product of an exponential orthogonal code and a Hadamard matrix. In this section we consider replacing this with a different inner code, still based on the Hadamard matrix. Our design is based on the following theorem.

Theorem 2: Given any three rows $\{r_1, r_2, r_3\}$ in a Hadamard matrix H_n with order $n, 4 \leq n \leq 32$, and representing the bitwise minority of the three rows by y , there exists one and only one row $x \in H_n \setminus \{r_1, r_2, r_3\}$, such that $y = x$.

In this theorem, the notation $H_n \setminus \{r_1, r_2, r_3\}$, means the matrix H_n excluding the rows listed in the set. Although we do not provide a formal proof of this theorem, the reader can confirm its correctness through exhaustive evaluation of all possible combinations of three rows in the Hadamard matrices of order 4, 8, 16, and 32.

If we use a single Hadamard matrix as the inner code, then the result of a 3-colluder minority attack on symbol position j will match, exactly, one symbol that is not involved in the collusion (assuming that the three colluding symbols differ). We'll refer to this matched symbol as the framed symbol. In fact, since the rows of the Hadamard matrix are orthogonal, the result will be orthogonal to all of the rows in the matrix except the framed symbol. The only non-zero entry in column j of the detection matrix would be the entry at the framed symbol – including the entries for the three participating symbols. Note also that different colluder sets may frame different innocent symbols.

Based on this observation, we design an inner code using the Hadamard matrix and its row-permuted versions. Consider the following inner code matrix G defined as

$$G = [H^{(1)} H^{(2)} H^{(3)} \dots H^{(K)}], \quad (2)$$

where $H^{(k)}$ is a randomly row-permuted version of a Hadamard matrix of order $n, 1 \leq k \leq K$.

Before continuing, we introduce some more notations. Each row of the matrix G corresponds to the codeword for a symbol. We refer to row i of G as f_i . This vector has length l and is composed of K codes, each of length n , which are rows from the differently permuted versions of the $n \times n$ Hadamard matrix. We refer to each of the K subcodes that comprise f_i as $f_{i,k}$ such that $f_i = [f_{i,1} \ f_{i,2} \ f_{i,3} \ \dots \ f_{i,K}]$. $f_{i,k}$ is the i^{th} row of $H^{(k)}$. As used previously, y is a code extracted from the captured work and $y^{(j)}$ is the portion of that code associated with symbol position j . This vector has length l and is composed of K codes, each of length n . Each of the K subcodes that comprise $y^{(j)}$ will be denoted $y_k^{(j)}$ such that $y^{(j)} = [y_1^{(j)} y_2^{(j)} y_3^{(j)} \dots y_K^{(j)}]$. With this, we note that

$$\langle y^{(j)}, f_i \rangle = \sum_{k=1}^K \langle y_k^{(j)}, f_{i,k} \rangle.$$

Consider a 3-colluder minority attack at symbol position j and assume that the three colluding symbols are all different. Let $y^{(j)}$ be the result of this collusion. For each $H^{(k)}$, a given innocent symbol subcode, $f_{i,k}$, has a $1/(n-3)$ chance of being framed (recall that $H^{(k)}$ has n rows, 3 of which are involved in the collusion). If K is large, we can expect every innocent symbol subcode to be framed $K/(n-3)$ times among all $H^{(k)}$ s, $1 \leq k \leq K$. The colluded subcode, $y_k^{(j)}$, will match exactly the framed subcode and will be orthogonal to all other subcodes. When $y_k^{(j)}$ and $f_{i,k}$ match, their correlation will be n and when they do not match, the correlation will be 0. This assumes that $\langle f_{i,k}, f_{i,k} \rangle = n$ (which is easily confirmed since $f_{i,k}$ contains n elements that are either +1 or -1.) So, for an innocent symbol, f_i , the expected value of $\langle y^{(j)}, f_i \rangle$ across all columns of G is $Kn/(n-3)$.

Recall that in the detection statistic of Equation (1), the correlation across the entire, length l inner code word, is normalized by the norm of the code word. Using the rows of the matrix G as the inner code, we see that $l = Kn$. Thus, the expected value of the detection statistic for an innocent symbol is $Kn/((n-3)\sqrt{l}) = \sqrt{l}/(n-3)$. Summarizing, at symbol position j , colluder symbols can be distinguished from innocent symbols by the difference in the detection statistics:

$$T_i^{(j)} = \begin{cases} 0, & \text{for colluder symbols,} \\ \sqrt{l}/(n-3), & \text{for innocent symbols.} \end{cases}$$

A detection scheme that exploits this difference has the potential to achieve resistance to a 3-colluder minority attack that will improve with the square root of the length of the inner code. In the next section we develop such a detection scheme.

4.2 Forensic detector

Using the matrix G of Equation (2) as an inner code we first analyze the output of the detection statistic introduced in Equation (1). We will then build on this to develop a new detection measure.

Due to the added noise in BSC, we may not have the exact correlation value as examined above. Thus we discuss the behavior of the detection statistics from expectation perspective. Let $\mu_c^{(j)}$ and $\mu_u^{(j)}$ be the expectations of the detection statistics, $T_i^{(j)}$, of colluders and innocent users at the j^{th} symbol position, respectively. We assume that three colluders are mounting a minority attack. There are three possibilities: the colluders have three different symbols, and thus three different inner codewords; two of the three colluders have the same inner code word; or all three colluders have the same inner codeword.

According to the analysis in Section 4.1, if the three colluders have three different inner codewords at the j^{th} symbol, then $\mu_c^{(j)} = 0$, and $\mu_u^{(j)} = \sqrt{l}/(n-3)$ for users with innocent symbols at position j and $\mu_u^{(j)} = 0$ for users with contributing symbols. When the three colluders share the same inner code f_i at the j^{th} symbol position, then $y^{(j)} = f_i$ and $\mu_c^{(j)} = \sqrt{l}$. In this case, an innocent user sharing this symbol would have $\mu_u^{(j)} = \sqrt{l}$, else $\mu_u^{(j)} = 0$. In the third scenario, two of the three colluders share the same symbol at position j . These two colluders will have the same bit value in each bit position of the inner code. The third colluder may also have this same bit value or may have the opposite bit value; each case occurring with equal probability. When the three bits values are the same, that bit value will appear in the colluded

copy. When the bit value differs, it will be the third colluder's bit value that will be the minority and will appear in the colluded copy. Taken together, we see that the inner code that results from this collusion will be the same as that of the third colluder. Thus $y^{(j)} = f_i$ for one of the three colluders ($\mu_c^{(j)} = \sqrt{l}$), and is orthogonal to f_i for the remaining two ($\mu_c^{(j)} = 0$). The case for the innocent users is the same as in the second case.

These results are summarized in **Table 1** below. In a minority collusion attack, a single symbol will fall into one of these three cases, but the next symbol may fall into a different case. It is likely that all three cases will occur.

Case	Colluders	Innocent Users
Colluders have 3 different symbols	$\mu_c^{(j)} = 0$	(symbol match) $\mu_u^{(j)} = 0$
		(no symbol match) $\mu_u^{(j)} = \sqrt{l}/(n-3)$
2 colluders have the same symbol	(majority) $\mu_c^{(j)} = 0$	(no symbol match) $\mu_u^{(j)} = 0$
	(minority) $\mu_c^{(j)} = \sqrt{l}$	(symbol match) $\mu_u^{(j)} = \sqrt{l}$
All colluders have the same symbol	$\mu_c^{(j)} = \sqrt{l}$	(no symbol match) $\mu_u^{(j)} = 0$
		(symbol match) $\mu_u^{(j)} = \sqrt{l}$

Table 1. Summary of the mean detection statistic at symbol position j for two sets of users, the colluders and the innocent users, under three different cases when three colluders apply a minority collusion attack on an inner code. In the first case, a symbol match occurs when the innocent user has a symbol that matches that of one of the three colluders. In the second case, a symbol match occurs when the innocent user has the symbol that matches the minority colluder. In the third case, a symbol match occurs when the innocent user has the same symbol as the three colluders.

4.2.1 Modified detection statistic

The observations of **Table 1** show that for the last case, all 3 colluders with the same symbol, $\mu_c^{(j)} \geq \mu_u^{(j)}$. However, in the first case, the opposite is true. In the second case, both $\mu_c^{(j)} \geq \mu_u^{(j)}$ and $\mu_c^{(j)} \leq \mu_u^{(j)}$ can happen. Each case has some likelihood of occurring at each symbol position. As a result, colluders do not necessarily have higher detection values than innocents and thus we cannot guarantee the accuracy of the maximum detector described in Section 2.2.

We would like to define a new detection statistic that makes it easy to distinguish between those users in the collusion set and those in the set of innocent users for all symbols under the minority attack. Of course, we need this to hold under random and majority interleaving attacks as well. Assuming that the alphabet size, q , is reasonably large (at least 8) to accommodate a non-trivial number of users in the system, an innocent user is much more likely to have symbols that fall into the “no symbol match” category in **Table 1**. At a given symbol position, the detection values, $T_i^{(j)}$, for innocent users will tend to form one cluster and those for colluders will tend to form another. If we knew the expected detection value of an innocent symbol, we could look at the difference between the detection value of each user and that of the known innocent symbol. This value should be small for innocent users and larger for colluders.

While we don't know the expected detection value of an innocent symbol, we expect the detection values for all symbols to form two clusters with the cluster of values from the innocent symbols having more members. Therefore, the median detection value from this set of detection values for all symbols is likely to be that of an innocent symbol.

Let $T_{median}^{(j)}$ be the median value of $G \times y^{(j)T}$, where $y^{(j)T}$ is the transpose of the row vector containing the colluded codeword at symbol position j and G is the inner codebook as defined in Section 4.1. $T_{median}^{(j)}$ is likely to be the detection value of an innocent inner codeword. We define a new detection statistic, $\tilde{T}_i^{(j)}$, as follows:

$$\tilde{T}_i^{(j)} = \left| \frac{\langle y^{(j)}, x_i^{(j)} \rangle - T_{\text{median}}^{(j)}}{\sqrt{l}} \right| \quad (3)$$

Example:

Consider the Hadamard matrix of order $n=8$ and an inner code composed of 5 row-permuted copies of this matrix.

Let $G = [H^{(1)} H^{(2)} H^{(3)} H^{(4)} H^{(5)}]$. This inner code has length $l = 40$ and supports $q = 8$ symbols. Assume that the first three symbols (rows) are used in a minority attack. In this example, we assume no additional noise. In order to explore this example, we again use the notation introduced in Section 4.1. The inner product $G \times y^{(j)T}$ is shown below

$$G \times y^{(j)T} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} \\ & & \vdots & & \\ f_{81} & f_{82} & f_{83} & f_{84} & f_{85} \end{bmatrix} \begin{bmatrix} y_1^{(j)} \\ y_2^{(j)} \\ y_3^{(j)} \\ y_4^{(j)} \\ \vdots \\ y_5^{(j)} \end{bmatrix}$$

where $f_{i,k}$ is the i^{th} row of $H^{(k)}$. The three colluding symbols are different, $f_1, f_2,$ and f_3 , so the expected correlations in all subcodes of the first 3 rows are 0. Most of the subcode correlations in the bottom 5 rows are also zero except for exactly one framed subcode in each column. The correlation of the framed subcode will be 8 because $n=8$. The subcode correlations are shown in the matrix of Figure 3. The inner product $G \times y^{(j)T}$ is the sum of each row, shown just to the right of the matrix. The median of this column, $T_{\text{median}}^{(j)}$, is 8 and it comes from innocent codeword. Thus, by subtracting $T_{\text{median}}^{(j)}$ from $G \times y^{(j)T}$, normalizing by \sqrt{l} , and taking the absolute value, $\tilde{T}_i^{(j)}$ becomes 0 for innocent users, while $\tilde{T}_i^{(j)} = 8/\sqrt{40}$ for colluders.

	$H^{(1)} \times y^{(j)T}$	$H^{(5)} \times y^{(j)T}$	$G \times y^{(j)T}$	$\tilde{T}_i^{(j)}$
Colluding Symbols	0	0	0	0	$8/\sqrt{40}$
	0	0	0	0	$8/\sqrt{40}$
	0	0	0	0	$8/\sqrt{40}$
Innocent Symbols	0	0	8	8	0
	0	0	0	8	0
	8	0	0	8	0
	0	0	8	8	0
	0	8	0	8	0

Figure 3. An example calculation of $\tilde{T}_i^{(j)}$. Shown is a matrix containing the values of subcode correlations in the calculation of $G \times y^{(j)T}$, the sum of each row to get $G \times y^{(j)T}$, and, in the far right column, the new detection values, $\tilde{T}_i^{(j)}$. In this example, the first three rows represent the colluding symbols and the detection values allow a clear classification of symbols as colluders or innocent symbols.

A similar analysis can be done for the case where the three colluders share the same inner code at symbol position j . Here, the colluding symbol will have a $G \times y^{(j)T} = 40$ and the correlation for all other symbols will be 0. The median will be 0 and $\tilde{T}_i^{(j)} = \sqrt{40}$. The third case, 2 of the 3 colluders share the same symbol, gives the same result.

4.2.2 Hard detector

The soft decoder of Section 2.2 gives equal weight to the detections at each symbol position. While the colluding symbols always have a higher detection value than the innocent symbols, an innocent symbol at one symbol location can have a larger detection value than a colluding symbol in another depending on how many of the colluders have the same value at the various symbol positions. This suggests that use of a “hard” detector may be more appropriate.

Our approach is to identify suspicious symbols at each symbol position. These are symbols most likely to be involved in a collusion attack. Then calculate the number of suspicious symbols for each user and choose the user with the largest number of suspicious symbols as the colluder. A suspicious symbol is identified as follows:

For the j^{th} symbol position, calculate $\tilde{T}_i^{(j)}$ for each symbol. As previously mentioned, we expect these detection values to form clusters. Suspicious symbols are those that fall “reasonably far” from the majority cluster. To find these, we sort the set of detection values $\{\tilde{T}_1^{(j)} \tilde{T}_2^{(j)} \dots \tilde{T}_q^{(j)}\}$ in descending order to get a vector Z . Then find the maximum of $Z(k)-Z(k-1) \forall 2 \leq k \leq q$, where $Z(k)$ is the k^{th} element of the sorted Z . Let the k_{max} be the index that gives the maximum of $Z(k)-Z(k-1)$. If this maximum is less than a threshold, i.e. $Z(k_{\text{max}})-Z(k_{\text{max}}-1) \leq \text{TH}$, then no symbols can be labeled suspicious at this symbol position and the symbol position is labeled “vague.” On the other hand, if the maximum exceeds a threshold, $Z(k_{\text{max}})-Z(k_{\text{max}}-1) > \text{TH}$, then each symbol i is declared as suspicious symbol if $T_i^{(j)} \geq Z(k_{\text{max}})$. Here, TH is a user-defined threshold and can be different for different symbol positions.

4.2.3 Adaptive detector

Although the hard detector works well under the minority attack for a small number of colluders, it is not as robust as the soft detector of Section 2.2 under other collusion attacks, such as the majority attack and the interleaving attack, when the number of colluders increases. Therefore, we propose to allow the forensic detector to adaptively switch between the hard detector and the soft detector.

Note that the hard detector highly depends on whether the suspicious inner codes of each symbol are detected successfully. When the number of vague symbols becomes large the hard detector will not work well. The soft detector, however, can continue to perform well because it retains more information. The adaptive detector will switch between soft detection and hard detection based on the number of vague symbols. If the number of vague symbols is smaller than a threshold TH_A , hard detection will be used. Otherwise, soft detection will be employed. TH_A is the threshold set by the user and should be adjusted according to the expected number of colluders and code parameters. **Error! Reference source not found.** illustrates the proposed adaptive detector.

5. EXPERIMENTAL RESULTS

In our simulation, there are 2^{20} users and the overall code length is around 6×10^6 . While this may seem long, it can be robustly embedded in a 30-second video host signal with current state-of-the-art video watermarking techniques. We choose a Reed-Solomon code with alphabet size 32 as the outer code. This outer code has code length 31 and minimum distance 28. The inner codes are the row-permuted Hadamard codes as presented in Section 4.1 and it has a length of 2×10^5 bits. The adaptive detector described in Section 4.2.3 is employed with TH for the hard detector set to 1.5 and TH_A for the adaptive detector set to 9. The BER for the BSC noise source varies from 0 to 0.32. The results presented here are based on 200 simulation iterations for each test scenario.

The first scenario involved a 3-colluder minority attack with BER=0. This is the case that caused trouble for the previous scheme and which motivated this current work. In 200 simulations, the detection error rate was 0. In every case, the user identified as a colluder, was in fact a colluder. BER was then increased in steps of 8% up to 32% with 200 simulations performed at each step. In this range of BER, the detection error rate remained 0.

These tests were repeated for a larger collusion set. In theory, we did not expect the resistance to minority attack to be sensitive to the number of colluders (for small sets of colluders). We increased the number of colluders in steps of 1 up to a maximum of 20 colluders. In this range, the detection error rate remained 0.

We then repeated all of the previously performed tests under the majority attack. Previously published works have shown the general ECC approach to be robust to majority attack and the purpose of these tests were to confirm that the

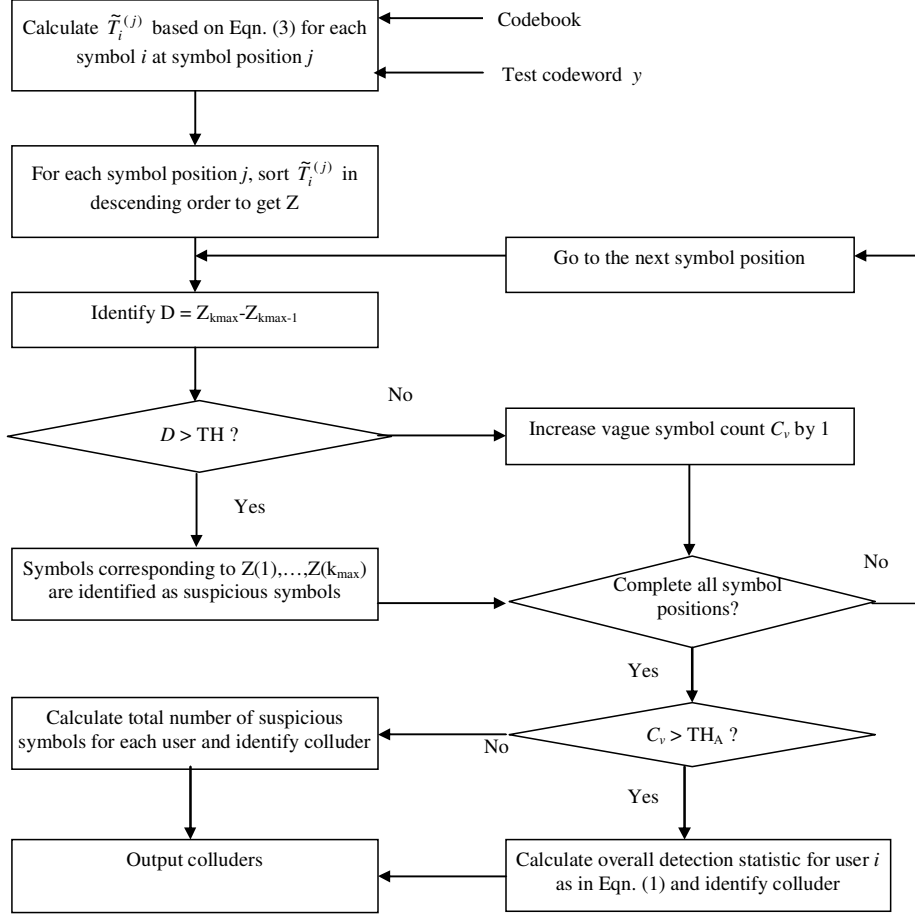


Figure 4 Adaptive Detector

improved scheme maintained this property. For the number of colluders in the range of 3 to 20 and for BER in the range of 0 to 32%, the detection error remained 0.

Finally, we repeated the tests under the random interleaving attack. Again, previously published works have shown the general ECC approach to be robust to this attack. For the number of colluders in the range of 2 to 20 and for BER in the range of 0 to 32%, the detection error remained 0.

6. CONCLUSION

In this paper, we presented a row-permuted orthogonal inner code construction along with an adaptive detector for ECC-based forensic marking. This code is designed to resist a minority collusion attack. The proposed scheme increases the collusion resistance of the ECC-based binary forensic code by one order of magnitude, from 2 to at least 20, under minority attack. The adaptive detector also helps maintain the detection performance of ECC-based forensic code against other collusion attacks, such as majority and random interleaving attacks. Simulation results show that the improved ECC-based binary forensic code, designed for 1 million users, can successfully detect up to 20 colluders mounting a majority, minority, or random interleaving attack followed by BSC with BER up to 32%.

REFERENCE

- [1] S. He and M. Wu, "Joint coding and embedding techniques for multimedia fingerprinting," *IEEE Trans. on Information Forensics and Security*, vol. 1, no.2, pp. 231-247, June. 2006.
- [2] H. V. Zhao, M.Wu, Z. J.Wang, and K. J. R. Liu, "Forensic analysis of nonlinear collusion attacks for multimedia fingerprinting," *IEEE Trans. Image Process.*, vol. 14, no. 5, pp. 646–661, May 2005.
- [3] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1897–1905, Sep.1998.
- [4] G. Tardos, "Optimal probabilistic fingerprint codes", in *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 2003, pp. 116–140.
- [5] A. Barg, G. R. Blakley, and G. Kabatiansky, "Digital fingerprinting codes: problem statements, constructions, identification of traitors," *IEEE Trans. Inform. Theory*, vol. 49, no. 4, pp. 852–865, Apr. 2003.
- [6] H. G. Schaathun, "Attack Analysis for He and Wu's Joint Watermarking /Fingerprinting Scheme", *IWDW*, Guangzhou, China, Dec. 2007.
- [7] S. W. Lin, S. He and J. A. Bloom, "Performance Study and Improvement on ECC-based Binary Anti-Collusion Forensic Code for Multimedia", submitted to *IEEE International Conf. on Acoustic, Speech, and Signal Processing (ICASSP'09)*.