

Performance Study and Improvement on ECC-based Binary Anti-Collusion Forensic Code for Multimedia

W. Sabrina Lin
ECE Department
University of Maryland
College Park, MD USA
wylin@umd.edu

Shan He
Thomson Corporate Research
Princeton, NJ, USA
Shan.He@thomson.net

Jeffrey Bloom
Dialogic Research Inc.
Eatontown, NJ, USA
bloom@ieee.org

ABSTRACT

Digital forensic coding is an emerging technology that offers proactive post-delivery protection of multimedia. Multi-user collusion attacks are powerful attacks against digital forensic marking, where groups of attackers collectively mount attacks to attenuate the identifying marks. The colluders usually perform post-processing, such as compression, on the colluded multimedia content before redistribution. The post-processing may introduce errors in the detected bits of the forensic code and represents a new challenge to the forensic code design. In this paper, we first study the performance of a simple extension from ECC-based forensic marking to create a binary forensic code, and then improve its performance to provide post-processing resistance. This new code is constructed by concatenating an orthogonal binary code with a large-distance Reed-Solomon code. The simulation results show that our binary forensic code design can achieve 100% detection rate when the number of colluders is less than 20 and up to 32% of the colluded forensic code is changed during the post-processing.

Categories and Subject Descriptors

K.4.4 [Computers and Society]: Electronic Commerce – Intellectual Property.

General Terms

Algorithms, Design, Security.

Keywords

Collusion Resistance, Collusion Attack, Multimedia, ECC-based Code, Binary Forensic Code

1. INTRODUCTION

Recent developments in network technologies and multimedia processing have facilitated the distribution and sharing of multimedia through networks. It is important to protect multimedia from illegal alteration, repackaging, and unauthorized redistribution. Digital forensic marking is an emerging tool that

offers proactive post-delivery protection of copyrighted content. It labels each distributed copy with the corresponding user's forensic mark, which can be used to trace traitors who illegally re-distribute their copies. Multi-user collusion attacks are cost-effective attacks against digital forensic marking, where several users, who are the colluders, combine information from different copies to generate a new copy in which the original forensic marks are removed or attenuated.

In the literature, there have been several works addressing collusion-resistant marking design. The Anti Collusion Code (ACC) proposed in [1] was the first anti-collusion forensic code designed specifically for multimedia content and it was based on combinatorial theories with a joint coding and embedding framework. ACC employs coded modulation using Gaussian orthogonal spreading sequences to modulate the code derived from balanced incomplete block design. The ECC-based forensic code in [2] uses Gaussian sequences to modulate symbols in the codeword and applies additive spread spectrum embedding [3]. On the other hand, there have been some works on designing forensic code for generic data. Boneh and Shaw [4] introduced the concept of a marking assumption, with which the colluders cannot change the bits that their marks all share. Based on this assumption, the Boneh-Shaw code uses a two-level binary code construction to resist collusion. Tardos [5] later proposed an optimal probabilistic-based binary code that reaches the lowest known bound $O(c^2 \log(n/\epsilon))$ for forensic code length. These works on forensic marking code for generic data assume a perfect channel between the colluded copy and the mark detector. However, when the protected data is multimedia, the colluders usually apply post-processing after collusion that forms an erroneous channel. For instance, the colluders can compress the multimedia to reduce the data size to efficiently redistribute the colluded copy. Therefore, it is important to design a collusion-resistant forensic code that is robust to channel error.

In this paper, we first study the performance of binary forensic codes under a binary symmetric channel (BSC) which is used to model the effect of multimedia post-processing on the forensic code. Specifically, we examine the performance of the ECC-based scheme and the Tardos code. To make the ECC based forensic marking scheme comparable to the Tardos code, we retain the ECC outer code level of [2] and replace the inner level with a binary bit string. We compare the traitor-tracing performance and resource consumption of the Tardos code and the ECC code under different types of collusion attacks. We find that the Tardos code outperforms the ECC code, but it has much higher resource consumption. We then propose an improved ECC

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM&Sec '09, September 7-8, 2009, Princeton, New Jersey, USA.

Copyright 2009 ACM 978-1-60558-492-8/09/09...\$10.00.

code that achieves comparable performance as the Tardos code and maintains the low computational resource consumption.

2. PERFORMANCE COMPARISON OF TARDOS CODE AND ECC CODE

In this section, we first briefly summarize the construction and detection of Tardos codes and ECC-based codes. Then we compare the performance of these two codes in a BSC under several collusion attacks.

2.1 Tardos Code

Let ε be the probability for each innocent user to be falsely accused as a colluder and let c be the maximum number of colluders the forensic system can resist. The Tardos code has code length $m = 100c^2 \log(1/\varepsilon)$ and can be generated as follows: The i^{th} bit of any user j 's codeword, X_{ji} , is independently generated with probability

$$P[X_{ji} = 1] = p_i. \quad (1)$$

Here, $p_i, 1 \leq i \leq m$ are *i.i.d.* random variables and $p_i = \sin^2 r_i$, where r_i are *i.i.d.* random variables uniformly distributed within $[t', \pi/2 - t']$ with $0 < t' < \pi/4$, $t' = \sin 2t$, $t = 1/300c$. Let y be the forensic code extracted from the colluded copy, then during detection, user j is accused if

$$\sum_{i=1}^m y_i U_{ij} > 20c \log(1/\varepsilon), \quad (2)$$

where $U_{ij} = \sqrt{1 - p_i / p_i}$ if $X_{ij} = 1$, and $U_{ij} = -\sqrt{p_i / (1 - p_i)}$ if $X_{ij} = 0$. See [5] for derivation of these equations.

2.2 ECC Code

The first step of generating an ECC-based binary forensic code is to construct an ECC outer code, such as a Reed-Solomon code, for N_u users with L symbols using an alphabet of size q , $\{f_0, f_1, \dots, f_{q-1}\}$. Then generate q orthogonal binary inner codes with l bits to represent the q symbols. The inner code and outer code are concatenated by replacing f_k with the k^{th} orthogonal inner codeword. The last step is to randomly permute each forensic codeword to prevent the code structure from being identified by the colluders [2].

Our initial study employs exponential orthogonal inner codes which are designed to preserve the colluders' information as much as possible. The orthogonal inner codes are constructed as follows: for the i^{th} codeword f_{i-1} , the first 2^{q-i} bits are 1s and the next 2^{q-i} are -1s; then repeat the same code 2^{i-1} times, ending up with 2^q bits. The columns of this exponential orthogonal inner code consist of all 2^q possible combinations of 1s and -1s, one column corresponding to the bits from q codewords at one bit position.

ECC-based code detection employs a correlation detector. Let y be the forensic code extracted from the colluded copy, $x^{(j)}$ be the forensic code of user j , and J be the set of all users. The detection statistics for user j is

$$TN^{(j)} = \frac{\langle y, x^{(j)} \rangle}{\|x^{(j)}\|}. \quad (3)$$

And, for a *maximum detector* strategy, user j is accused as a colluder if $TN^{(j)} \geq TN^{(i)} \forall i \in J$. Under this maximum detector, the probability of falsely accusing an innocent user (P_{fa}) for the ECC code equals $1 -$ the probability of detection (P_d) since the detector will always accuse one user. To achieve a fair comparison with the Tardos code, we employ the threshold detector which accuses user j if

$$TN^{(j)} \geq TN^{(i)} \quad \forall i \in J, \text{ and } TN^{(j)} \geq Th(\delta),$$

where $Th(\delta)$ is the threshold when $BER = \delta$. The $Th(\delta)$ is set by experiments to ensure $P_{fa} \leq \varepsilon$.

2.3 Performance Evaluation

2.3.1 Collusion attacks

In this paper, we focus on the two most common attacks for multimedia forensic codes—the averaging/majority attack and the interleaving attack.

We typically assume one of two cases regarding the knowledge of the attackers. In one case, the attackers know the marking space. The marking space is a projection of the pixel space where the watermark bits are mutually exclusive (i.e., they do not overlap) See [6] for a detailed discussion of Marking Spaces. With knowledge of the marking space, the attackers can perform an interleaving attack. The interleaving attack is implied in most of the forensic code works [4][5]. During interleaving collusion, colluders contribute their copies frame by frame with approximately equal share. At the bit level with c colluders, the interleaving attack works as randomly choosing a colluder's fingerprint code with probability $1/c$ for each bit.

In the second case, we assume that the attacker does not know the marking space. Here, we assume that the collusion takes place in the pixel domain or common frequency domain. The averaging attack is widely studied in the literature of multimedia forensic marking due to its ease of implementation. In this attack, the colluders simply add and then average their copies pixel by pixel or component by component. If an additive embedding layer is employed, the averaging operation is also reflected in the code-bit domain by mapping the averaged values to -1 or 1 of the code bit. This is equivalent to a majority interleaving attack, where the colluders who are aware of the marking space pick the bit that appears in a majority of their copies.

2.3.2 Simulation settings

In our simulation, there are around 1 million (2^{20}) users, 5 colluders, and $\varepsilon = 10^{-3}$ for the Tardos code. The BER of the BSC varies from 0 to 0.32. The simulation results are based on 200 simulation runs. As discussed in [5], Tardos codes can be extended to $1/(1-2\delta)^2$ times longer in order to resist a BER of δ . As a result, the code length of the Tardos code, resistant to a BER of 0.32, is 5×10^5 . The employed ECC inner code needs 2^q bits, in which q should be a power of 2. Therefore, the largest q we can use is 16, and due to the RS code constraint $L \leq q$, we take $L = 15$. As a result, the ECC-based design leads to 10^6 bits. Therefore, to

give a fair comparison, we repeat the Tardos code twice to arrive at the same code length.

2.3.3 Traitor-tracing performance

Figure 1 shows the detection rate P_d of catching one colluder and the false accusation rate P_{fa} of the ECC code and the Tardos code under interleaving and averaging/majority attacks. We experimentally set the threshold $\text{Th}(\delta)$ for the ECC code to maintain the same P_{fa} as the Tardos code. It is clear that the Tardos code achieves perfect detection under both types of attack for a BER up to 0.32, at which the ECC code only has a detection rate around 20%.

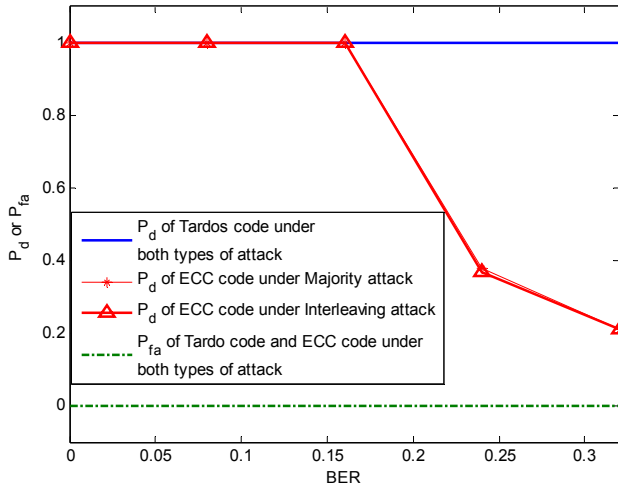


Figure 1: Detection rate for the Tardos code and the ECC-based code under the interleaving attack and the averaging attack with $P_{fa} = 0$.

2.3.4 Resource consumptions

We further analyze the computational complexity of the Tardos code and ECC-based code during code generation and detection. There are two ways to obtain the forensic code for each user during embedding/detection. We could either generate the watermark code on the fly or retrieve the code from memory where the code has been pre-generated and stored.

• Generating the forensic code

Tardos code: Generating the Tardos code involves two steps: obtaining probability p for each bit and generating the codeword for each user. The first step needs m' operations and the second step requires $N_u m'$ operations, where $m' = 100c^2 \log(N_u/\epsilon)/(1-2\delta)^2$ and N_u is the number of users. The overall complexity would be $O(N_u m') = O(100c^2 N_u \log(N_u/\epsilon)/(1-2\delta)^2)$. With our simulation settings, the overall complexity necessary to generate the Tardos codes is $5 \times 10^5 \times 2^{20} = 5 \times 10^{11}$.

ECC-based code: The inner code of the ECC-based code is fixed; q codewords each with length of 2^q . Thus we need $q \times 2^q$ operations to generate the inner code. The computational complexity of generating the outer RS code is $O(kLq^k)$, which is $O(kq^{k+1})$ given $L \leq q$ and $k = \sqrt[q]{N_u}$. The overall complexity of generating the ECC code becomes $O(q2^q + kq^{k+1})$. In our settings,

with $q = 16$, $k = 5$, we have the complexity for ECC code generation as $2^4 \times 2^{16+5} \times 2^{24} = 8.5 \times 10^7$, which is 4 orders of magnitude lower than that of the Tardos codes.

• Storage

Tardos code: Since Tardos codes are probabilistic forensic codes and there is no code structure, the code bits for all the users must be stored. Thus, we need $N_u m' = 2^{20} \times 2^{10} = 2^{39}$ bits = 64 GB disk space.

ECC-based code: ECC-based codes are the concatenation of an inner code and an outer code. By saving the inner codes and outer codes, we can easily construct the forensic codes for all the users. As a result, for ECC-based codes, the disk space needed is $q2^q$ for the inner code and $LN_u \log_2 q$ for the outer code, which is $2^4 \times 2^{16} + 2^4 \times 2^{20} \times 4 \approx 2^{26}$ bits = 8 MB in our experiment settings. We can see that to store the codebook, ECC-based code only requires 1/8000 of Tardos code space.

• Detection

For a Tardos code, the correlation is performed for every user's codeword. Thus, the overall detection complexity is $O(N_u L_{all})$, where L_{all} is the overall code length. For an ECC-based code, the decoding complexity is $O(\sqrt[k]{N_u} L_{all})$ based on the analysis in [2].

Thus, the decoding complexity of ECC-based code is much lower than Tardos code.

Note that the above analysis is for Tardos code originally proposed in [5], following which there have been many improvements in terms of code length [7] and memory consumption [8]. However, the improvement on the code length, e.g. 5 times shorter in [7], is not significant enough to reduce the complexity and storage cost of the Tardos codes to be comparable to ECC-based codes. The memory consumption studied in the literature, e.g. [8], is different from our analysis in that it is the extra memory for generating and storing the continuous distribution of the probability used to generate the code. While our analysis is on two types of resource consumptions: the computational complexity, rather than the extra memory cost, in generating the Tardos code, and the storage resources of the code itself if the code is generated beforehand.

As a result, we can see that the Tardos code has high generation and detection computational complexity and storage consumption, which makes the Tardos code not suitable for applications where computation power and storage space are limited. On the other hand, the ECC coding requires significantly lower computation and storage resources and is promising for practical applications. In the following section, we will focus on improving the traitor tracing performance of ECC-based coding while keeping the advantage of low resource consumption.

3. IMPROVED ECC-BASED CODE

In this section, we propose an improved ECC-based code by reducing the code correlation.

3.1 Reducing code correlation

It has been shown that the overall forensic marking correlation plays an important role in the collusion resistance performance of

a Gaussian-based ECC forensic marking scheme [2]. When the modulation sequence becomes a binary orthogonal code, it is still true that lower correlation leads to higher collusion resistance (we omit the proof in this paper due to space limitations). Thus we can improve the traitor tracing performance of ECC-based codes by reducing the overall code correlation. Since the ECC-based code studied in this paper employs orthogonal inner codes, the overall correlation of the code is determined by the parameters of the RS outer code. The correlation between RS code words is $(L-k+1)/L$. Since $L \leq q$ and $k = \sqrt[q]{N_u}$, reducing code correlation can be achieved by increasing alphabet size q .

We can double the alphabet size of the RS code to 32 and choose $k=4$ and the outer code length 31. Under this setting, the overall correlation becomes $3/31$. Compared with the code we used in Section 2.3, where the RS code had an alphabet size of 16, dimension 5, and code length 15, the correlation is reduced from 0.27 to 0.1. However, if we still employ the orthogonal binary inner code as described in Section 2.2, the overall code length would be increased to $31 \times 2^{32} = 1.3 \times 10^{11}$, which is much larger than the original code length of 10^6 . To maintain the same code length, the inner code with alphabet size 32 should be no longer than 3.2×10^4 bits. Therefore, we need to design orthogonal inner codes with much shorter code length.

3.2 Building short orthogonal inner codes

It is easy to show that the Kronecker product of two orthogonal binary codes taking value 1 or -1 is an orthogonal code. Here, the Kronecker product of X and Y , denoted by $X \otimes Y$, is defined by replacing all bit 1s in X by Y , and all bit -1s in X by $-Y$. In order to shorten the code length of the inner code, we could use a short orthogonal code as the inner code or combine a low-order exponential code with a short orthogonal code. One example of the binary orthogonal code with short length is the Hadamard code. The Hadamard code can be viewed as a $q_c \times q_c$ orthogonal matrix which exists when $q_c = 2^m$, $m \in \mathbb{N}$. A Hadamard matrix H_n with order n can be generated recursively by

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}, \text{ where } H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

One possibility is to use the Hadamard matrix itself as the inner code or the Kronecker product of two Hadamard matrices. However, our study shows that the performance of the Hadamard matrix alone or the combined inner code from two Hadamard matrices are not good because the columns of the code contain very few possible combinations of binary bits among codes, and thus do not preserve much information about the colluders. In our improved inner code design, we employ the exponential orthogonal code as one of the two orthogonal codes for Kronecker product to keep as much of the colluders' information as possible. The other code can be Hadamard matrix or an exponential orthogonal code with shorter length. Our study shows that both codes lead to similar performance. In our following results section, we use the Kronecker product of an exponential code with a Hadamard matrix. We choose the Hadamard matrix with order n , H_n and the exponential inner code with order q/n , and the final inner-code length would be $n \times 2^{q/n}$. Since in our setting, the

required code length of inner codes is 3.2×10^4 , we choose $n = 4$, $q/n = 8$ and repeat the concatenated code 2^5 times.

4. SIMULATION RESULTS

In this section, we show the performance of the Tardos code and the improved ECC-based code with the maximum threshold detector discussed in Section 2.2. The simulation settings are the same as Section 2.3: the total number of users is around 1 million, and the probability of false accusation for the Tardos code is set to be 10^{-3} . The Tardos code is designed to resist BER up to 0.32 and up to 5 colluders. The improved ECC code uses 32 symbols with outer-code length 31, and the Kronecker product of Hadamard matrix of order 4 with exponential orthogonal code of order 8 as the inner code. As illustrated in Figure 2, the ECC code achieves 100% detection rate as does the Tardos code when there are 5 colluders. Thus by reducing the code correlation, the collusion resistance of ECC-based code has been significantly improved.

We also increased the colluder number to be more than the designed value to explore the performance limit of the Tardos code and the improved ECC-based code. As shown in Figure 3, at $\text{BER} = 0.32$, the detection rates of both codes start to drop when there are more than 18 colluders. When the colluder number reaches 21, the detection rate of the Tardos code drops to zero and the improved ECC code still has 60% chance of catching one colluder.

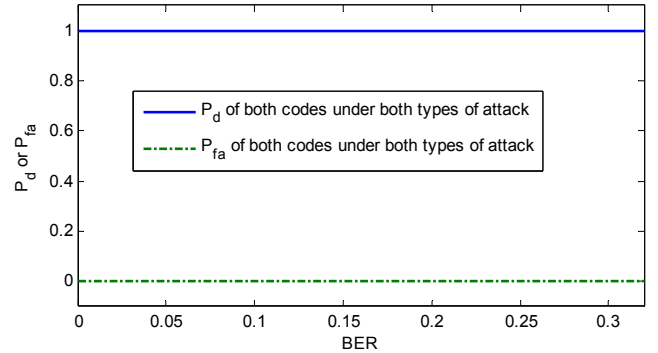


Figure 2: Detection rates for the Tardos code and the improved ECC-based code under interleaving attack and averaging attack with 5 colluders as a function of BER.

Furthermore, we compare the resource consumptions of the Tardos code and the improved ECC-based code. Following the same analysis as in Section 2.3.4, the storage and computation consumptions of Tardos code and the improved ECC-based code are listed in Table 1. From Table 1, Figure 2, and Figure 3, we can see that a Tardos code designed to resist up to 5 colluders and BER of 0.32 uses more than 3 thousand times more resources than the improved ECC-based code while both codes achieve 100% detection rate under the test conditions against majority and interleaving attacks. When the actual number of colluders is larger than what the code is designed for, the performance of the improved ECC-based code degrades more gracefully.

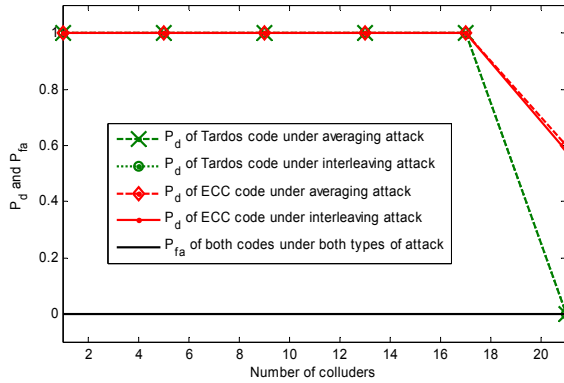


Figure 3: Detection rates for the Tardos code and the improved ECC-based code under interleaving and averaging attacks with BER = 0.32 as the number of colluders increases.

	Tardos	Improved ECC
Code Generation (unit operator)	5×10^{11}	1.3×10^8
Storage	64 GB	20 MB
Detection	$O(N_u L_{all})$	$O(\sqrt[k]{N_u} L_{all})$

Table 1: Resource consumptions of the Tardos code and the improved ECC code.

5. CONCLUSION

In this paper, we studied the collusion resistance performance of the Tardos code and the ECC-based code under BSC with various BERs. Results show that although Tardos code provides 100% detection rate within the examined BER and colluder number range, it requires at least 4 orders of magnitude more resource consumption than an equivalent ECC-based code. We then designed an improved ECC-based code by reducing the code

correlation. Results show that for around 1 million users, the improved ECC-based code can achieve 100% detect rate on catching one colluder when there are more than one and half dozen colluders mounting interleaving or averaging/majority attacks, while maintaining the advantages over Tardos codes in lower computational complexity or storage cost. In our future work, we will examine the performance of the proposed code when applied onto video signals.

6. REFERENCES

- [1] W. Trappe, M. Wu, J. Wang, and K. J. R. Liu, "Anti-collusion fingerprinting for multimedia," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 1069–1087, Apr. 2003
- [2] 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.
- [3] I. Cox, J. Killian, F. Leighton, and T. Shamoan, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.
- [4] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1897–1905, Sep.1998.
- [5] G. Tardos, "Optimal probabilistic fingerprint codes", In proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 116–140.
- [6] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*, Morgan Kaufmann Publishers, Inc., San Francisco, 2001.
- [7] Skoric, B. Vladimirova, T.U. Celik, M. Talstra, J.C. , "Tardos fingerprinting is better than we thought," *IEEE Transactions on Information Theory*, Aug. 2008, pp. 3663-3676.
- [8] Nuida, K., Hagiwara, M., Watanabe, H., Imai, H.: Optimization of Tardos' fingerprinting codes in a viewpoint of memory amount. In: Proceedings of 9th Information Hiding (IH 2007), LNCS 4567, pp. 279–293, 2007.