

Collusion-Secure Fingerprinting for Digital Data

(Extended Abstract)

Dan Boneh*
dabo@cs.princeton.edu

James Shaw
jhs@cs.princeton.edu

Department of Computer Science
Princeton University
Princeton, NJ 08544

Abstract. This paper discusses methods for assigning codewords for the purpose of fingerprinting digital data (e.g., software, documents, and images). Fingerprinting consists of uniquely marking and registering each copy of the data. This marking allows a distributor to detect any unauthorized copy and trace it back to the user. This threat of detection will deter users from releasing unauthorized copies.

A problem arises when users collude: For digital data, two different fingerprinted objects can be compared and the differences between them detected. Hence, a set of users can collude to detect the location of the fingerprint. They can then alter the fingerprint to mask their identities. We present a general fingerprinting solution which is secure in the context of collusion. In addition, we discuss methods for distributing fingerprinted data.

1 Introduction

The human fingerprint has been used to distinguish individuals since the turn of the century. Wishing to distinguish other objects, people have applied fingerprint-like identifiers to a variety of objects from maps and mathematical tables to diamonds and explosives. With the increasing use of digital computers and data, there has been a strong desire to fingerprint these data as well. This paper discusses a general method to fingerprint digital data. We use the following terminology:

A *mark* is one bit of information which is encoded in an object. For instance, in a book, the use of one word versus another in some point in the text, can be used as a mark.

A *fingerprint* is a collection of marks. Thus, a fingerprint can be thought of as a binary word of length L , where L is the number of marks in the object.

A *distributor* is the sole supplier of fingerprinted objects.

A *user* is the registered owner of a fingerprinted object.

The core idea of fingerprinting is that each user receives a copy of the object containing a unique fingerprint. This fingerprint can be used to identify the user.

* Supported in part by NSF CCR-9304718.

Because a fingerprinted object can be traced, the users will be deterred from releasing an unauthorized copy. The users will wish to destroy the fingerprint, while the distributor will wish to prevent this from happening.

Unlike human fingerprints, digitally-stored data can be altered with no harm to the object. It is the goal of the distributor to make fingerprints which are unalterable and undetectable. However, for digital data, the fingerprints of two users can be easily compared and the differences between them detected. Two users can collude to produce a new fingerprint, potentially masking their identities, unless this is explicitly prevented. To the best of our knowledge, previous work does not discuss fingerprints which resist these effects of collusion. We discuss fingerprints which can be traced back to their users even if multiple users collude. Our fingerprinting method can be applied to any form of digital data. A related result by Chor, Fiat and Naor [5] discusses a similar problem in a different setting. Within the paper, we use as an example the specific problem of software protection.

The Software Publishers Association estimates that in one year over seven billion dollars was lost due to software piracy: the illegal copying and redistribution of computer software [12]. A variety of copy protection schemes exist in an attempt to slow this piracy. However, most of these schemes lack rigor and are eventually bypassed. A rigorous treatment was given by Goldreich and Ostrovsky. They discuss how to protect software using an oblivious RAM which prevents an opponent from gaining any information about a running program [8]. The main difference between such schemes and fingerprinting is that fingerprinting does not actively prevent copying. Instead it is used as a deterrent. Fingerprinting does not require any special hardware and has negligible effect on the running time.

By definition, each fingerprinted copy of an object is unique; however, sending each user a different copy may not be practical. The distribution problem is the avoidance of giving each user a different copy. By using redundancy between copies, we show how to reduce the cost of distributing fingerprinted data.

Our fingerprint consists of a set of binary marks². We denote the two states of a mark by $\{0, 1\}$. By colluding, users can detect a specific mark if it differs between their copies; otherwise a mark can not be detected. The main property the marks should satisfy is that users can not change the state of an undetected mark without rendering the object useless. A detected mark's state can be changed to any one of the three states $\{0, 1, ?\}$. The symbol '?' means that the mark is unreadable, i.e. is impossible to determine if the mark is 0 or 1.

We assume that marks satisfying these properties exist for the objects being fingerprinted. We refer to this as the Marking Assumption for which a precise definition is given in the next section. Note that if there is no collusion, by the Marking Assumption, fingerprinting is trivial: the fingerprint assigned to each user will be the user's serial number.

There has been much research investigating the Marking Assumption in a

² For simplicity, we use binary marks throughout the paper. Using a larger fixed size alphabet will change our results by constant factors.

variety of domains. Wagner [15] gives a taxonomy of fingerprints and suggests subtle marks for computer software. Marks have been embedded in digital images [13, 4], in documents [3], and in computer programs [7]. In all of these domains, our scheme allows these marks to be combined to form secure fingerprints. Thus our results are general, allowing a variety of digital data to be fingerprinted.

The remainder of the paper is organized into the following sections: Section 2 defines codes which prevent a group of users from framing another user; we call such codes c -frameproof. Section 3 defines more general codes which enable the distributor to trace a copy in the presence of c users colluding; we refer to these codes as c -secure codes. A construction of c -secure codes is presented in Section 4. Section 5 describes an efficient method for distributing fingerprinted data. Section 6 concludes with a summary and discussion of open problems.

Throughout the paper, the following notation will be used. Given an l -bit word $w \in \{0, 1\}^l$ and a set $I = \{i_1, \dots, i_r\} \subseteq \{1, \dots, l\}$ we denote by $w|_I$ the word $w_{i_1} w_{i_2} \dots w_{i_r}$, where w_i is the i 'th bit of w . We refer to $w|_I$ as the *restriction* of w to the bit positions in I .

2 Protection Against Naive Redistribution

In order to protect its product, a distributor could uniquely mark each copy, using marks that are undetectable by individual users. The general strategy used by the distributor is to assign a unique codeword to each user and to ship each user his data marked with that codeword. If we assume that users can not change the marks, then if an unauthorized copy is found, the codeword embedded in it must belong to some user u . Unfortunately, the assumption that users can not change marks is unrealistic. A coalition of users can detect some of the marks, namely the ones where their copies differ. They can then change these marks arbitrarily. Henceforth, we assume that this is the only type of mark which users can detect. The following definitions formally capture this notion.

Definition 1. A set $\Gamma = \{w^{(1)}, \dots, w^{(n)}\} \subseteq \{0, 1\}^l$ will be called an (l, n) -code. The codeword $w^{(i)}$ will be assigned to user u_i , for $1 \leq i \leq n$. We refer to the set of words in Γ as the **codebook**.

Definition 2. Let $\Gamma = \{w^{(1)}, \dots, w^{(n)}\}$ be an (l, n) -code and C be a coalition of users. For $i \in \{1, \dots, l\}$ we say that bit position i is **undetectable** for C if the words assigned to users in C match in their i 'th bit. Formally, suppose $C = \{u_1, \dots, u_c\}$. Then bit i is undetectable if $w_i^{(u_1)} = w_i^{(u_2)} = \dots = w_i^{(u_c)}$.

Definition 3. Let $\Gamma = \{w^{(1)}, \dots, w^{(n)}\}$ be an (l, n) -code and C be a coalition of users. Let R be the set of undetectable bit positions for C . Define the **feasible set** of C as

$$F(C; \Gamma) = \left\{ w \in \{0, 1\}^l \text{ s.t. } w|_R = w^{(u)}|_R \right\}$$

for some user u in C . Thus the feasible set contains all words which match the coalition's undetectable bits. Usually we omit the Γ and denote $F(C; \Gamma)$ by $F(C)$.

Formally, the Marking Assumption discussed in the introduction states that any coalition of c users is only capable of creating an object whose fingerprint lies in the feasible set of the coalition. If a coalition changes one of the detectable marks to be unreadable, then we arbitrarily say that the mark is set to 0.

In this section we consider the simple problem of *naive redistribution* which occurs when a user redistributes his copy of the object without altering it. If an unauthorized copy of the object is found containing user u 's codeword we would like to say that user u is guilty. However, u could claim that he was framed by a coalition who created an object containing his codeword. Thus, we would like to construct codes that satisfy the following property: no coalition can collude to frame a user not in the coalition. We usually relax this condition by limiting the size of the coalition to c users. We call such codes c -frameproof codes.

If the code used to fingerprint the object is kept hidden from the users, then the construction of frameproof codes becomes trivial: to every one of the n users assign a unique codeword chosen at random. A coalition of users can not frame a user not in the coalition since they do not know his codeword. We would like to construct codes that are c -frameproof even if the codebook is known to the users. This requirement can be formally stated as follows:

Definition 4. A code Γ will be called a c -frameproof code if every set $W \subset \Gamma$, of size at most c , satisfies $F(W) \cap \Gamma = W$.

The definition states that in a c -frameproof code, the only codewords in the feasible set of a coalition of at most c users are the codewords of members of the coalition. Thus, no coalition of at most c users can frame a user who is not a member of the coalition. It is interesting to note that for random codes the length of the code must be exponential in c . Otherwise, a coalition of c users is likely to detect all the bits.

2.1 Construction of c -frameproof codes

We now show a construction for c -frameproof codes. A similar idea to the one presented below will be used in Section 4 for the construction of more powerful codes. We begin by introducing a simple (n, n) -code which is n -frameproof. Define the code $\Gamma_0(n)$ to be the (n, n) -code containing all n -bit binary words with exactly one 1. For example, the code $\Gamma_0(3)$ for three users is $\{100, 010, 001\}$. The following claim is immediate.

Claim 5. $\Gamma_0(n)$ is a (n, n) -code which is n -frameproof.

It is not difficult to see that any n -frameproof code must have length at least n . Hence, the code length of Γ_0 is optimal. The length of Γ_0 is linear in the number of users and is therefore impractical. We will use the code Γ_0 to construct shorter codes. We first recall some basic definitions from the theory of error correcting codes. See [14] for more details.

Definition 6. A set, \mathcal{C} , of N words of length L over an alphabet of p letters is said to be an $(L, N, D)_p$ -Error Correcting Code or in short, an $(L, N, D)_p$ -ECC, if the Hamming distance between every pair of words in \mathcal{C} is at least D .

The idea of this construction is to compose the code $\Gamma_0(n)$ with an error correcting code. Let $\Gamma = \{w^{(1)}, \dots, w^{(n)}\}$ be an (l, p) -code and let \mathcal{C} be a $(L, N, D)_p$ -ECC. We denote the composition of Γ and \mathcal{C} by Γ' . The code Γ' is an (lL, N) -code defined as follows: for a codeword $v = v_1 v_2 \dots v_L \in \mathcal{C}$ let

$$W_v = w^{(v_1)} || w^{(v_2)} || \dots || w^{(v_L)}$$

where $||$ means concatenation of strings. The code Γ' is the set of all words W_v , i.e. $\Gamma' = \{W_v \mid v \in \mathcal{C}\}$.

Lemma 7. Let Γ be a c -frameproof (l, p) -code and \mathcal{C} be a $(L, N, D)_p$ -ECC. Let Γ' be the composition of Γ and \mathcal{C} . Then Γ' is a c -frameproof code, provided $D > L(1 - \frac{1}{c})$.

Proof. Let C be a coalition of c users. We show that $F(C; \Gamma')$ contains no words from Γ' other than those of C . Let $v^1, \dots, v^c \in \mathcal{C}$ be the codewords of \mathcal{C} from which the codewords of the coalition C were derived.

Assume towards a contradiction that $F(C; \Gamma')$ contains a word $W \in \{0, 1\}^{lL}$ which belongs to a user $u \notin C$. Let $z \in \mathcal{C}$ be the codeword from which W was derived. For all $k = 1, \dots, c$, the words z and v^k match in less than L/c positions. This follows since the minimal Hamming distance of \mathcal{C} is bigger than $L(1 - 1/c)$. Hence, there must exist a position $1 \leq j \leq L$ for which $z_j \neq v_j^k$ for all $k = 1, \dots, c$.

Let $C_j = \{w^{(v_j^1)}, \dots, w^{(v_j^c)}\}$. Since Γ is a c -frameproof code we know that $w^{(z_j)}$ is not in $F(C_j; \Gamma)$. Since $w^{(z_j)}$ is a subword of W , this implies that $W \notin F(C; \Gamma')$. This contradiction proves the lemma. \square

We note that the condition that \mathcal{C} has a large minimal distance can be relaxed. To make the proof work it suffices to require that no set of c words of \mathcal{C} "cover" a word of \mathcal{C} outside the set. This property has been studied in [6]. Using this relaxed requirement does not improve the constructions.

The error correcting codes we are using have large minimal distance and hence, low rate. By picking the codewords randomly it is possible to obtain a good low rate code. We state this in the following lemma, which is immediate from the Chernoff bound [2, Appendix A].

Lemma 8. For any positive integers p, N let $L = 8p \log N$. Then there exists a $(L, N, D)_{2p}$ -ECC where $D > L(1 - \frac{1}{p})$.

The main theorem of this section now follows.

Theorem 9. For any integers $n, c > 0$ let $l = 16c^2 \log n$. Then there exist an (l, n) -code which is c -frameproof.

Proof. By Lemma 8 we know that there exists an error correcting code with parameters $(L, n, L(1 - 1/c))_{2c}$ for $L = 8c \log n$. Combining this with the code $\Gamma_0(2c)$ and Lemma 7 we get a c -frameproof code for n users whose length is $2cL = 16c^2 \log n$. \square

To make this construction explicit we must use an explicit low rate error correcting code. Explicit constructions of such codes are described in [1]. The explicit construction are not as good as the bounds provided by Lemma 8. Using a simple explicit low rate code it is possible to obtain codes of length $l = c^2 \log^2(n)$.

3 c -secure Codes

Codes which are secure against naive redistribution do not provide enough information when users collude. Indeed, two users can collude to generate an object which is marked by a word which is not a codeword. In this section we address this more general problem.

Assume a distributor marks an object with a code Γ . Now, suppose a coalition of users, C , colludes to generate an unregistered object marked by some word x and then distributes this new object. When this object is found, the distributor would like to detect a subset of the coalition who created it.

We illustrate the problem with an example. Let Γ be some code. Let C_1 and C_2 be two coalitions of c users each such that $C_1 \cap C_2 = \emptyset$. Suppose an unregistered object is found which is marked by a codeword x which is feasible for both C_1 and C_2 . Then both coalitions are suspect. Since their intersection is empty, we can not determine with certainty who created the unregistered object. This situation should be avoided and therefore we require that when the intersection of C_1 and C_2 is empty, the intersection of their feasible sets $F(C_1)$ and $F(C_2)$ should also be empty. This is captured in the following definition.

Definition 10. A code Γ will be called **totally c -secure** if

$$C_1 \cap \dots \cap C_r = \emptyset \Rightarrow F(C_1) \cap \dots \cap F(C_r) = \emptyset .$$

for all coalitions C_1, \dots, C_r of at most c users each.

It seems that totally secure codes provide a good solution to the problem of collusion. Unfortunately, when $c > 1$, totally c -secure codes do not exist.

Theorem 11. For $c \geq 2$ and $n \geq 3$ there are no totally c -secure (l, n) -codes.

Proof. Clearly, it is enough to show that there are no totally 2-secure codes. Let Γ be an arbitrary (l, n) -code. Let w_1, w_2, w_3 be three distinct codewords assigned to users u_1, u_2, u_3 respectively. Let w be the bitwise majority of w_1, w_2, w_3 . The intersection of the coalitions $\{u_1, u_2\}, \{u_1, u_3\}, \{u_2, u_3\}$ is empty; however, the word w is feasible for all three coalitions. This proves that Γ is not totally 2-secure. \square

Note that the above proof relies on the fact that Γ is made up of binary strings. The proof can be generalized to show that there are no totally c -secure codes even when larger alphabets are used. This generalization is based on the fact that users can make a detectable mark unreadable, e.g. by removing it. This is the main difference between our model and the model used in [5].

The negative result proved in Theorem 11 forces us to weaken our requirements for marking schemes. We will allow the distributor to make some random choices when embedding the codewords in the objects. The point is that the random choices will be kept hidden from the users. This will enable us to construct codes which will capture a member of the guilty coalition with high probability.

Suppose a coalition C of c users creates an illegal copy of an object. Codes that enable the capture of a member of the coalition C with probability at least $1 - \epsilon$ are called c -secure codes with ϵ -error. Here the probability is taken over the random choices made by the distributor and the random choices made by the coalition C .

To simplify the exposition we don't give the formal combinatorial definition of c -secure codes. Instead we present a construction of a good c -secure code.

4 Construction of c -secure Codes

The idea for the construction c -secure codes is similar to the one used in Section 2.1. We first construct an (l, n) -code which is n -secure. Thus, no matter how large the coalition is, we will be able to trace an illegal copy back to a member of the coalition with high probability. The length of this code is $n^{O(1)}$ and hence, too large to be practical. We then show how this code can be used to construct c -secure codes for n users whose length is $\log^{O(1)}(n)$ when $c = O(\log n)$.

We begin by presenting an (l, n) -code which is n -secure with ϵ -error for any $\epsilon > 0$. Let c_m be a column of height n in which the first m bits are 1 and the rest are 0. The code $\Gamma_0(n, d)$ will consist of all columns c_1, \dots, c_{m-1} , each duplicated d times. The amount of duplication will determine the error probability ϵ . For example, the code $\Gamma_0(4, 3)$ for four users A, B, C, D is

```
A : 111111111
B : 000111111
C : 000000111
D : 000000000
```

Let $w^{(1)}, \dots, w^{(n)}$ denote the codewords of $\Gamma_0(n, d)$. Before the distributor embeds the codewords of $\Gamma_0(n, d)$ in an object he makes the following random choice: the distributor randomly picks a permutation³ $\pi \in S_l$. User u_i 's copy of the object will be fingerprinted using the word $\pi w^{(i)}$. Note that the same permutation π is used for all users. The point is that π will be kept hidden from

³ S_l is the full symmetric group of all permutations on l letters. For a word $x \in \{0, 1\}^l$ and a permutation $\pi \in S_l$ we denote by πx the l -bit word $x_{\pi(1)}x_{\pi(2)} \dots x_{\pi(l)}$.

the users. Keeping the permutation hidden from the users is equivalent to hiding the information of which mark in the object encodes which bit in the code. It is a bit surprising that this simple random action taken by the distributor is sufficient to overcome the barrier of Theorem 11 and enables us to prove the following theorem:

Theorem 12. For $n \geq 3$ and $\epsilon > 0$ let $d = 2n^2 \log(2n/\epsilon)$. The code $\Gamma_0(n, d)$ is n -secure with ϵ -error.

The length of this code is $d(n-1) = O(n^3 \log \frac{n}{\epsilon})$. To prove the theorem we must describe an algorithm, which given a word x generated⁴ by some coalition C , outputs a member of C with probability $1 - \epsilon$. First we introduce some notation.

1. Let B_m be the set of all bit positions in which the users see columns of type c_m . That is, B_m is the set of all bit positions in which the first m users see a 1 and the rest see a 0. The number of elements in B_m is d .
2. For $2 \leq s \leq n-1$ define $R_s = B_{s-1} \cup B_s$.
3. For a binary string x , let $\text{weight}(x)$ denote the number of 1's in x .

Before we describe the algorithm we give some intuition. Suppose user s is not a member of the coalition C which produced the word x . The hidden permutation π prevents the coalition from knowing which marks represent which bits in the code $\Gamma_0(n, d)$. The only information the coalition has is the value of the marks it can detect. Observe that without user s a coalition sees exactly the same values for all bit positions $i \in R_s$. For instance, in the code $\Gamma_0(4, 3)$ above, the coalition A, C, D sees the exact same bit pattern for all bit positions in R_2 . Hence, for a bit position $i \in R_s$, the coalition C can not tell if i lies in B_s or in B_{s-1} . This means that whichever strategy they use to set the bits of $x|_{R_s}$, the 1's in $x|_{R_s}$ will be roughly evenly distributed between $x|_{B_s}$ and $x|_{B_{s-1}}$ with high probability. Hence, if the 1's in $x|_{R_s}$ are not evenly distributed then, with high probability, user s is a member of the coalition that generated x .

Algorithm 1. Given a word $x \in \{0, 1\}^l$, find a subset of the coalition that produced x .

1. If $\text{weight}(x|_{B_1}) > 0$ then output "User 1 is guilty".
2. If $\text{weight}(x|_{B_{n-1}}) < d$ then output "User n is guilty".
3. for all $s = 2$ to $n-1$ do: Let $k = \text{weight}(x|_{R_s})$. If

$$\text{weight}(x|_{B_{s-1}}) < \frac{k}{2} - \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}}$$

then output "User s is guilty".

⁴ When we say that a coalition C generated a word x , we mean that the bits of x have already been unscrambled using π^{-1} . For example, the first bit of x is the value of the mark which encodes the first bit of the codewords.

The correctness of the algorithm is proved in the next two lemmas.

Lemma 13. *Consider the code $\Gamma_0(n, d)$ where $d = 2n^2 \log(2n/\epsilon)$. Let S be the set of users which algorithm 1 pronounces as guilty on input x . Then with probability at least $1 - \epsilon$, the set S is a subset of the coalition C that produced x .*

Proof. Suppose user 1 was pronounced guilty, i.e. $1 \in S$. Then $\text{weight}(x|_{B_1}) > 0$. This implies that user 1 must be a member of C (otherwise the bits in B_1 would be undetectable for C which would imply that $\text{weight}(x|_{B_1}) = 0$). Similarly if $n \in S$ then $n \in C$.

Suppose the algorithm pronounces user $1 < s < n$ as guilty. We show that the probability that s is not guilty is at most ϵ/n . This will show that the probability that there exists a user in S which is not guilty is at most ϵ .

Let s be an innocent user, i.e. $s \notin C$. As was discussed above, this means that the coalition C can not distinguish between the bit positions in R_s . Since the permutation π was chosen uniformly at random from the set of all permutations, the 1's in $x|_{R_s}$ may be regarded as being randomly placed in $x|_{R_s}$. Let $k = \text{weight}(x|_{R_s})$. Define Y to be a random variable which counts the number of 1's in $x|_{B_{s-1}}$ given that $x|_{R_s}$ contains k 1's. For any integer r in the appropriate range:

$$\Pr[Y = r] = \frac{\binom{M}{r} \binom{M}{k-r}}{\binom{2M}{k}}$$

where $M = 2n^2 \log(2n/\epsilon)$ is the size of B_{s-1} . Clearly, the expectation of Y is $k/2$. To bound the probability that s was pronounced guilty we need to bound

$$\Pr \left[Y < k/2 - \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}} \right]$$

from above. It can be shown that the variance of Y is $\text{var}(Y) = \frac{k}{4} \frac{(2M-k)}{(2M-1)}$.

Let X be a binomial random variable over k experiments with success probability $1/2$. The variance of X is $k/4$. We see that the variance of Y is smaller than the variance of X , implying that for any $a > 0$

$$\Pr \left[\left| Y - \frac{k}{2} \right| > a \right] \leq \Pr \left[\left| X - \frac{k}{2} \right| > a \right] \leq 2e^{-2a^2/k}$$

where the last inequality follows from the standard Chernoff bound [2, Appendix A]. Plugging in $a = \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}}$ leads to

$$\Pr \left[Y < \frac{k}{2} - \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}} \right] \leq \Pr \left[\left| Y - \frac{k}{2} \right| > \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}} \right] \leq 2e^{-\log(2n/\epsilon)} = \frac{\epsilon}{n} .$$

Thus, if user s is innocent then the probability of her being pronounced guilty by Algorithm 1 is at most ϵ/n . Therefore, the probability that some innocent user will be pronounced guilty is at most ϵ . This proves the lemma. \square

Lemma 14. Consider the code $\Gamma_0(n, d)$ where $d = 2n^2 \log(2n/\epsilon)$. Let S be the set of users which algorithm 1 pronounces as guilty on input x . Then the set S is not empty.

The proof of the lemma relies on the following claim.

Claim 15. Suppose the set S is empty. Then for all s we have

$$\text{weight}(x|_{B_s}) \leq 2s^2 \log(2n/\epsilon) .$$

Proof. The proof is by induction on s . For $s = 1$, the claim is immediate since if user 1 is not guilty then $\text{weight}(x|_{B_1}) = 0$.

Now, we assume the claim holds for $s < n - 1$ and prove it for $s + 1$. Define

$$k = \text{weight}(x|_{B_s}) \quad ; \quad k' = \text{weight}(x|_{B_{s+1}}) \quad ; \quad t = \text{weight}(x|_{R_{s+1}}) .$$

Then the following three conditions are satisfied:

$$t = k + k' \quad ; \quad k \leq 2s^2 \log(2n/\epsilon) \quad ; \quad k \geq \frac{t}{2} - \sqrt{\frac{t}{2} \log \frac{2n}{\epsilon}}$$

The first condition follows from the fact that $R_{s+1} = B_s \cup B_{s+1}$. The second is the inductive hypothesis and the third follows from the fact that user s was not pronounced guilty, i.e. $s \notin S$. We show that these three conditions imply $k' \leq 2(s+1)^2 \log(2n/\epsilon)$ which will prove the claim for $s+1$.

$$\begin{aligned} k' &= t - k \leq \frac{t}{2} + \sqrt{\frac{t}{2} \log \frac{2n}{\epsilon}} = \frac{k + k'}{2} + \sqrt{\frac{1}{2}(k + k') \log \frac{2n}{\epsilon}} \\ &\leq \frac{2s^2 \log(2n/\epsilon) + k'}{2} + \sqrt{\frac{1}{2}(2s^2 \log \frac{2n}{\epsilon} + k') \log \frac{2n}{\epsilon}} \end{aligned}$$

which leads to

$$k' \leq 2s^2 \log \frac{2n}{\epsilon} + \sqrt{2(2s^2 \log \frac{2n}{\epsilon} + k') \log \frac{2n}{\epsilon}} .$$

Suppose $k' = 2r^2 \log(2n/\epsilon)$ for some constant r . Substituting for k' and dividing by $2 \log(2n/\epsilon)$ we get

$$r^2 \leq s^2 + \sqrt{s^2 + r^2} .$$

It is not difficult to see that for this inequality to be satisfied when $s \geq 1$ we must have $r \leq s + 1$. Hence, $k' \leq 2(s+1)^2 \log(2n/\epsilon)$. \square

Proof of Lemma 14. Suppose S is empty. Since user n was not pronounced guilty we know that $\text{weight}(x|_{B_{n-1}}) = d = 2n^2 \log(2n/\epsilon)$. On the other hand, for $s = n - 1$ Claim 15 shows that $\text{weight}(x|_{B_{n-1}}) \leq 2(n-1)^2 \log(2n/\epsilon)$. This contradiction proves the lemma. \square

This completes the proof of Theorem 12. The following theorem provides a weak lower bound on the length of n -secure codes. We leave the proof for the final version of the paper.

Theorem 16. Any (l, n) -code which is n -secure with ϵ -error must have length $l \geq n - 1$, for any $\epsilon > 0$.

4.1 Logarithmic length c -secure codes

The n -secure code constructed in the previous section enables us to use the techniques of [5] to construct c -secure (n, l) -codes of length $l = c^{O(1)} \log n$. We thank Naor [10] for pointing out this relation. In this section we demonstrate how to apply the simplest technique from [5] to construct a short c -secure code from the n -secure code of the previous section. The basic idea is to use the n -secure code as the alphabet over which the techniques of [5] can be applied.

Let \mathcal{C} be an (L, N) -code over an alphabet of size n where the codewords are chosen independently and uniformly at random⁵. The idea is to compose our n -secure code $\Gamma_0(n, d)$ with the code \mathcal{C} as we did in the proof of Lemma 7. We call the resulting code $\Gamma'(L, N, n, d)$. Thus, the code $\Gamma'(L, N, n, d)$ contains N codewords and has length $Ld(n - 1)$. It is made up of L copies of $\Gamma_0(n, d)$. We will refer to these copies as the *components* of $\Gamma'(L, N, n, d)$. The point is that the codewords of the code \mathcal{C} will be kept hidden from the users. This is in addition to keeping hidden the L permutations used when embedding the L copies of $\Gamma_0(n, d)$ in the object.

Theorem 17. *Given integers N, c and $\epsilon > 0$ set $n = 2c$; $L = 2c \log(2N/\epsilon)$ and $d = 2n^2 \log(4nL/\epsilon)$. Then, $\Gamma'(L, N, n, d)$ is a code which is c -secure with ϵ -error. The code contains N words and has length $l = O(Ldn) = O(c^4 \log(N/\epsilon) \log(1/\epsilon))$.*

To prove the theorem we show an algorithm that finds a member of the guilty coalition and then prove it's correctness.

Algorithm 2. Given a word $x \in \{0, 1\}^l$, find a subset of the coalition that produced x .

1. Apply Algorithm 1 to each of the L components of x . For each component $i = 1, \dots, L$ arbitrarily choose one of the outputs of Algorithm 1. Set y_i to be this chosen output. Note that y_i is a number between 1 and n . Next, form the word $y = y_1 \dots y_L$.
2. Find the word $w \in \mathcal{C}$ which matches y in the most number of position (ties are broken arbitrarily).
3. Let u be the user whose codeword is derived from $w \in \mathcal{C}$. output "User u is guilty".

Lemma 18. *Let x be a word which was produced by a coalition C of at most c users. Then with parameters as in Theorem 17, Algorithm 2 will output a member of C with probability at least $1 - \epsilon$.*

Proof (sketch). Let W be the set of codewords in \mathcal{C} that correspond to the users in the coalition C . For every $1 \leq i \leq L$ Algorithm 1 guarantees that y_i will match w_i for some $w \in W$ with probability $1 - \epsilon/2L$. This follows from the choice of d and the fact that in component i the users in C see words from

⁵ In [5] the codewords of \mathcal{C} are considered as random hash functions

$h : \{1, \dots, L\} \rightarrow \{1, \dots, n\}$.

$\Gamma_0(n, d)$. It follows that the above condition will be satisfied in *every* component with probability at least $1 - \epsilon/2$. We refer to this as event A .

Recall that the size of W is at most c . Therefore, when event A occurs there must exist a word $w \in W$ which matches y in L/c positions. However, since the words in \mathcal{C} are random and hidden from the users, any word in \mathcal{C} which is not in W is expected to match y in only $L/n = L/2c$ positions. Using the Chernoff bound it can be shown that the probability that a random word will match y in L/c positions is less than $\epsilon/2N$. Hence, the probability the some word in $\mathcal{C} \setminus W$ will match y in L/c positions is at most $\epsilon/2$. This shows that when event A occurs, the algorithm will output a member of C with probability at least $1 - \epsilon/2$. Combining this with the fact that event A occurs with probability at least $1 - \epsilon/2$ proves the lemma. \square

5 Distribution Scheme

Up until now, we have been ignoring distribution of the uniquely fingerprinted copies. This is fair, as at worst we can send each user an entire unique copy. However, this is impractical for products such as electronic books, software or CD-ROMs which are mass produced. We would like to come up with a scheme in which a user receives a bulk of data common to all users, and a small number of extra bits unique to him. We refer to the bulk of data as the *public data* and denote it by D . We refer to the extra bits given privately to user u as the *private data* and denote it by M_u . For the distribution scheme to be secure, given (D, M_u) , user u should not be able to deduce any information about the fingerprints in copies given to other users.

Throughout this section, let Γ be an (l, n) -code which is used to fingerprint the objects. We denote the object to be distributed by P and let L be its length. Assume the object P can be partitioned into l pieces with exactly one mark in each piece. Let $p(r, s)$ be the r th piece which contains the r th mark in state $s \in \{0, 1\}$. For any r , the pieces $p(r, 0)$ and $p(r, 1)$ are interchangeable, that is a copy with one replaced with the other will behave identically. Given a codeword $x = x_1 x_2 \dots x_l \in \Gamma$, let $P(x) = \{p(1, x_1), p(2, x_2), \dots, p(l, x_l)\}$ be the partition set implied by x .

Theorem 19. *It is possible to solve the distribution problem using $O(L)$ size public data and $O(L^{1/m}l/\log L)$ private data for some fixed constant m .*

Proof (sketch). We encode each piece $p(r, s)$ with a secure private key cryptosystem. See [9] for the precise definitions; such systems are known to exist under some standard hardness assumptions. Let $f_k : \{0, 1\}^{L'} \rightarrow \{0, 1\}^{L'}$ be such a system, where $L' = L/l$ is the length of a single piece $p(r, s)$. The key k will have length $K = (L')^{1/m}$ for some fixed constant m . The fact that f_k is a secure private key cryptosystem implies that for a random key k , given $f_k(x)$, no polynomial time predicate can extract one bit of information about x with non negligible probability. This property is crucial for the security of our distribution scheme.

For each piece, $p(r, s)$, pick a random key $k(r, s)$. The public data will be

$$D = \{f_{k(r,s)}(p(r, s)) \mid 1 \leq r \leq l, s \in \{0, 1\}\} .$$

The size of D is $2L$, twice the size of the original object. Let $x \in \Gamma$ be the word associated with user u . The private data given to user u is the collection of keys necessary to decrypt his pieces:

$$M_u = \{k(1, x_1), k(2, x_2), \dots, k(l, x_l)\} .$$

Using this scheme, given (D, M_u) , any user u can construct a usable copy of the distributed object. The size of the private data, M_u is $lK = O(l^{1-1/m} L^{1/m}) = O(lL^{1/m})$.

It is possible to further reduce the size of the private data by using double encryption: we group the keys $k(r, s)$ into blocks. Each block is encrypted and made public. Each user will receive keys that will enable him to decrypt his block of keys. This method requires public data of size $O(L)$ and private data of size $O(L^{1/m} \frac{l}{\log L})$ for some constant $m > 1$. \square

It is worth noting that when implementing this distribution scheme, one can use a standard private key cryptosystem such as DES. Such systems use fixed length keys. This leads to private data of length $O(l/\log(L))$.

6 Discussion and Open Problems

The most significant contribution of this paper is to show how to overcome collusion when fingerprinting digital data. To summarize our results, we restrict the size of coalitions to be at most $\log n$ where n is the number of users. For the problem of naive redistribution considered in Section 2, we constructed codes of length $\log^3 n$. For the general redistribution problem considered in Section 3 we constructed codes of length $\log^6 n$. Furthermore, we demonstrated an efficient method for shipping fingerprinted data which requires only a small constant factor increase in the size of the data.

There are still many open problems which remain to be solved. Recall that throughout the paper, we assumed that secure marks can be embedded in the fingerprinted data. A mark encodes one bit of information and is secure if it can only be detected by collusion. To emphasize the fact that we will not be dealing with the implementation of secure marks, we referred to the assumption that they exist as the "Marking Assumption". In many domains, one can construct secure marks with the aid of problems that are believed to be hard. For instance, when fingerprinting movies, a single mark can be encoded by using one camera view point versus another. The choice of one view point versus another in a specific scene, encodes one bit of information in the film. Given an image, the problem of transforming the image to an image taken from a different view point is believed to be hard. As this method of marking can be used to fingerprint movies, we say that the Marking Assumption holds in the domain of movies.

Showing that the Marking Assumption is satisfied for software is much harder. As was stated in the introduction, there is a great deal of empirical evidence to support the existence of secure marks in software. However, to the best of our knowledge, no formal results have been obtained.

To overcome the fact that secure marks are hard to construct we could make our codes resilient to noise. Suppose a coalition of users is able to change a small fraction of its undetectable bits. Is it possible to construct codes which are still c -secure?

Acknowledgments

The authors wish to thank Richard Lipton for some discussions on this work. We are grateful to Moni Naor for many suggestions and comments on this work.

References

1. N. Alon, J. Bruck, J. Naor, M. Naor and R. Roth, *Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs*, IEEE Transactions on Information Theory, vol. 38, 1992, pp. 509–516.
2. N. Alon and J. Spencer, *The probabilistic method*, Wiley, 1992.
3. J. Brassil, S. Low, N. Maxemchuk and L. O’Gorman, *Electronic marking and identification techniques to discourage document copying*, Proceedings of Infocom ’94, pp. 1278–1287, June 1994.
4. G. Caronni, *Assuring ownership rights for digital images*, H.H. Brueggemann and W. Gerhardt-Haeckl (Ed.) Proceedings of ‘reliable IT systems’ (verlaessliche IT-Systeme) VIS ’95 Vieweg Publishing Company, Germany, 1995.
5. B. Chor, A. Fiat and M. Naor, *Tracing traitors*, Proceedings of Crypto, 1994, pp. 257–270.
6. P. Erdős, P. Frankl, Z. Füredi, *Families of finite sets in which no set is covered by the union of r others*, Israel J. of math. 51, 1985, pp. 79–89.
7. D. Glover, *The protection of computer software*, Cambridge University, 2nd ed., 1992.
8. O. Goldreich and R. Ostrovsky, *Software protection and simulation on oblivious RAMs*, to appear in JASM.
9. S. Goldwasser, *The search for provably secure cryptosystems*, AMS Lecture notes cryptology and computational number theory, 1990.
10. M. Naor, *Private communications*.
11. B. Schneier, *Applied cryptography*, Wiley, 1994.
12. Software Publishers Association, Press release, 1994.
13. K. Tanaka, Y. Nakamura and K. Matsui. *Embedding secret information into a dithered multi-level image*, Proceedings of the 1990 IEEE Military Communications Conference, pp. 216–220, September 1990.
14. van Lint, *Introduction to coding theory*, Springer-Verlag, 1982.
15. N. Wagner, *Fingerprinting*, Proceedings of the 1983 IEEE Symposium on Security and Privacy, April, 1983, pp. 18–22.