Collusion-Secure Fingerprinting for Digital Data

Dan Boneh
dabo@cs.stanford.edu

Dept. of Computer Science
Stanford University
Stanford, CA, 94305-9045

James Shaw
jhs@cs.princeton.edu

Dept. 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, music, and video. 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

Fingerprinting is an old cryptographic technique. Several hundred years ago logarithm tables were protected by fingerprinting them. The idea was to introduce tiny errors in the insignificant digits (i.e. tenth digit right of the decimal point) of $\log x$ for a few random values of x. A different set of x's was chosen for each copy of the logarithm table. If an owner of a logarithm table ever sold illegal copies of it, the tiny errors in the table enabled the "police" to trace an illegal copy back to its owner. The owner would then be prosecuted and penalized in some way.

Nowadays no one is interested in protecting logarithm tables. However, the technique of fingerprinting is still in use. Examples include maps, diamonds and explosives. With the increasing popularity of digital data, there is a strong desire to fingerprint these data as well. Examples of digital data to which fingerprinting may apply include documents, images, movies, music and executables.

When fingerprinting digital data one must address the problem of collusion. For instance, suppose the logarithm table discussed above is stored as a file. Each user is given a slightly altered copy of the file. If two users get together they can easily run `diff` on their two files to discover all the locations where the files differ. This simple operation reveals the location of the hidden marks. The users can then remove these marks and resell their logarithm table without ever worrying about being caught. Notice that two users could only detect those marks in which their copies differ. They could not detect marks where both their copies agree. We intend to use this small amount of information which the two users could not remove to trace any copy they generate back to one of them.

Throughout the paper we use the following terminology: A *mark* is a position in the object which can be in one of s different states. For instance, in the logarithm table example, introducing an error in the value of $\log 3$ means that $\log 3$ is marked. If there are s error values used we say that the mark has s possible states. A *fingerprint* is a collection of marks. Thus, a fingerprint can be thought of as a word of length L over an alphabet Σ of size s. Here L is the number of marks embedded in the object. A *distributor* is the sole supplier of fingerprinted objects. A *user* is the registered owner of a fingerprinted object.

The process of fingerprinting an object involves assigning a unique codeword over Σ^L to each user. The user receives a copy of object with the marks set according to his assigned codeword. 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. 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 variety of domains. Wagner [22] gives a taxonomy of fingerprints and suggests subtle marks for computer software. Marks have been embedded in digital video [20, 6, 9, 10], in documents [5], and in computer programs [12]. In all these domains, our scheme allows these marks to be combined to form collusion resistant fingerprints. Thus our results are general, applying to a variety of digital data.

Previously a weaker model of collusion secure fingerprinting was studied in [4]. Our results are more efficient by an exponential amount both is terms of the number of users and in terms of the coalition size. The reason for this dramatic improvement is our use of *randomness*. We rely on randomness in two steps: one is in the construction and proof of security of our fingerprinting codes. The other is in the compostion of our construction with earlier elegant results of Chor, Fiat and Naor [8].

We note that several recent proposals enhance the functionality of fingerprinting schemes in various ways. Assymetric fingerprinting [15, 16, 3] ensures that a corrupt distributor cannot frame an inocent user. Anonymous fingerprinting [17] makes use of a registration service to eliminate the need for the distributor to keep detailed records binding codewords to users. The combinatorial properties required for collusion secure fingerprinting are further studied in [19, 7].

The paper is organized as follows. In Section 2 we introduce our notation and explicitely state the Marking Assumption. In Section 3 we discuss a naive scenario to familiarize the reader with the notation. In Section 4 we define our notion of collusion secure codes and in Section 5 we construct such codes. In Section 6 we give a lower bound on the length of collusion secure codes. Section 7 describes a scheme for distributing fingerprinted data on bulk media such as CD-ROMs.

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

2 Fingerprinting codes

We begin by defining some notation. From here on Σ will denote some alphabet of size s representing the s different states of the marks. The letters in Σ will be denoted by the integers 1 to s.

Definition 2.1 *A set* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\} \subseteq \Sigma^l$ *will be called an* (l,n)**-code**. *The codeword* $w^{(i)}$ *will be assigned to user* u_i, *for* $1 \le i \le n$. *We refer to the set of words in* Γ *as the* **codebook**.

Definition 2.2 *Let* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ *be an* (l,n)*-code and* C *be a coalition of users. For* $i \in \{1, \ldots, l\}$ *we say that position* i *is* **undetectable** *for* C *if the words assigned to users in* C *match in their* i*'th position. Formally, suppose* $C = \{u_1, \ldots, u_c\}$. *Then position* i *is undetectable if* $w_i^{(u_1)} = w_i^{(u_2)} = \ldots = w_i^{(u_c)}$.

As was discussed in the introduction our objective is to design a collusion-secure method of assigning codewords to users. Let C be a coalition of users. We must first characterize the set of objects which the coalition C can generate. Suppose the i'th mark is detectable by the coalition C. The coalition can generate an object in which the i'th mark is in any of its s states. Furthermore, the coalition can generate an object in which the mark is in an *unreadable* state. When the police recovers an illegal copy of the object it can not determine which state an unreadable mark is in. For instance, in the logarithm table example this would correspond to the coalition creating a table which does not contain the entry $\log 3$ where $\log 3$ is a detectable mark. We denote a mark in an unreadable state by '?'. The resulting set of codewords is called the *feasible* set of the coalition. Formally the feasible set is defined as follows:

Definition 2.3 *Let* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ *be an* (l,n)*-code and* C *be a coalition of users. Let* R *be the set of undetectable positions for* C. *Define the* **feasible set** *of* C *as*

$$F(C;\Gamma) = \left\{ w \in (\Sigma \cup \{?\})^l \;\; s.t. \;\; w|_R = w^{(u)}|_R \right\}$$

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* $F(C)$.

We will occasionally denote $\Sigma \cup \{?\}$ by Σ'. 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.

Example If two users A and B are assigned the codewords

$$\begin{array}{ccccc} 3 & 2 & 3 & 1 & 2 \\ 1 & 2 & 2 & 1 & 2 \end{array}$$

then their feasible set is $F(AB) = \Sigma' \cdot 2 \cdot \Sigma' \cdot 1 \cdot 2$.

3 Protection Against Naive Redistribution

To familiarize the reader with our notation we begin by considering a toy problem which we refer to as *naive redistribution*. Naive redistribution 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 3.1 *A code Γ is c-**frameproof** 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.

3.1 Construction of c-frameproof codes

We now show a construction for c-frameproof codes over the binary alphabet $\Sigma = \{0, 1\}$. First we introduce 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 3.1 *$\Gamma_0(n)$ 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. This follows since any coalition of $n-1$ users must not be able to detect at least one of the bit positions; otherwise, they could frame a user not in the coalition. Since there are n coalitions of size $n-1$, the code must have at least length 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 [21] for more details.

Definition 3.2 *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 $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ 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 $v = v_1 v_2 \ldots v_L \in \mathcal{C}$ let

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

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 3.2 *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 $D > L\left(1 - \frac{1}{c}\right)$.*

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 $v^1, \ldots, v^c \in \mathcal{C}$ 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 $k = 1, \ldots, c$, 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 $k = 1, \ldots, c$.

Let $C_j = \{w^{(v_j^1)}, \ldots, w^{(v_j^c)}\}$. 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 [11]. 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 3.3 *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 3.4 *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 3.3 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 3.2 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 3.3. Using a simple explicit low rate code it is possible to obtain codes of length $l = c^2 \log^2(n)$.

4 c-secure Codes

We now turn our attention to the full problem of collusion secure fingerprinting. Suppose 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. In other words, there must exist a *tracing algorithm* A which on input x output a member of the coalition C. For our purposes the tracing algorithm A may be regarded as a function $A : \{0,1\}^n \to \{1, \ldots, n\}$ where n in the number of users. This leads to the following definition:

Definition 4.1 *A code Γ is* **totally c-secure** *if there exists a tracing algorithm A satisfying the following condition: if a coalition C of at most c users generates a word x then $A(x) \in C$.*

The tracing algorithm A on input x must output a member of the coalition C that generated the word x. Hence, an illegal copy can be traced back to at least one member of the guilty coalition. Clearly there is no hope in recovering the entire coalition since some of its members might be passive; they are part of the coalition, but they contribute nothing to the construction of the illegal copy.

We now derive a necessary condition of a code to be totally c-secure. Consider the following scenario: 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, it is not possible to determine with certainty who created the unregistered object. It follows that if Γ is totally c-secure then when the intersection of C_1 and C_2 is empty, the intersection of their feasible sets $F(C_1)$ and $F(C_2)$ must also be empty. In general we obtain the following lemma:

Lemma 4.1 *If Γ is a totally c-secure code then*

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

for all coalitions $C_1, \ldots, C_r$ 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 4.2 *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. Define the majority word $M = \mathrm{MAJ}(w^{(1)}, w^{(2)}, w^{(3)})$ by

$$M_i = \begin{cases} w_i^{(1)} & \text{if} \quad w_i^{(1)} = w_i^{(2)} \quad \text{or} \quad w_i^{(1)} = w_i^{(3)} \\ w_i^{(2)} & \text{if} \quad w_i^{(2)} = w_i^{(3)} \\ ? & \quad \textit{otherwise} \end{cases}$$

One can readily verify that the majority word M is feasible for all three coalitions $\{u_1, u_2\}, \{u_1, u_3\}, \{u_2, u_3\}$. However the intersection of the coalitions is empty. Therefore by Lemma 4.1 the code Γ is not totally 2-secure. $\square$

The proof of the theorem shows that if a coalition employs the 'majority' strategy it is guaranteed to defeat all fingerprinting codes. Based on this result it seems that all is lost: fingerprinting is not possible in the presence of collusion. Fortunately there is a way out of this trap: *use randomness.*

Theorem 4.2 forces us to weaken our requirements for marking schemes. We intend to 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 enables us to construct codes which will capture a member of the guilty coalition with high probability.

An (l, n) fingerprinting scheme is a function $\Gamma(u, r)$ which maps a user number $1 \leq u \leq n$ and a string of random bits $r \in \{0, 1\}^*$ to a codeword in Σ^l. The random string r is the set of random bits used by the distributor and kept hidden from the users. We denote a fingerprinting scheme by Γ_r.

Suppose a coalition C of c users creates an illegal copy of an object. Fingerprinting schemes 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.

Definition 4.2 *A fingerprinting scheme Γ_r is c-secure with ϵ-error if there exists a tracing algorithm A satisfying the following condition: if a coalition C of at most c users generates a word x then*

$$\Pr[A(x) \in C] > 1 - \epsilon$$

where the probability is taken over the random bits r and the random choices made by the coalition.

The tracing algorithm A on input x outputs a member of the coalition C that generated the word x *with high probability*. With this definition at hand we turn to the construction of c-secure codes.

We point out that Chor, Fiat and Naor [8] considered a similar problem in an entirely different settings. In our terms their result enables one to construct c-secure codes under the assumption that marks can not become unreadable. Under this assumption one can even construct totally c-secure codes. Indeed, the proof of Theorem 4.2 relied on the existence of unreadable marks.

Unfortunately, in the context of fingerprinting the assumption that marks can not become unreadable is unrealistic. As was discussed in Section 2 once a coalition detects a mark, that mark can be made unreadable in various ways (recall that by unreadable we mean that when an illegal copy is discovered it is impossible to determine which state the mark is in). For this reason, the results of Chor, Fiat and Naor by themselves are insufficient for fingerprinting. However, as we shall see, they become quite useful when composed with our results described in the next section.

5 Construction of collusion secure codes

The idea for the construction c-secure codes is similar to the one used in Section 3.1. 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 $\Gamma_0(n, d)$ consists of all columns $c_1, \ldots, c_{n-1}$, each duplicated d times. The amount of duplication determines the error probability ϵ. For example, the code $\Gamma_0(4, 3)$ for four users A, B, C, D is

$$\begin{array}{ll} A: & 111111111 \\ B: & 000111111 \\ C: & 000000111 \\ D: & 000000000 \end{array}$$

Let $w^{(1)}, \ldots, w^{(n)}$ 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 permutation[1] $\pi \in S_l$. 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 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 4.2 and enables us to prove the following theorem:

Theorem 5.1 *For $n \geq 3$ and $\epsilon > 0$ let $d = 2n^2 \log(2n/\epsilon)$. The fingerprinting scheme $\Gamma_0(n, d)$ is n-secure with ϵ-error.*

[1] 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 $x_{\pi(1)} x_{\pi(2)} \cdots x_{\pi(l)}$.

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 generated[2] 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 $2 \le s \le n-1$ define $R_s = B_{s-1} \cup B_s$.

3. For a binary string x, let 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 $x \in \{0,1\}^l$, find a subset of the coalition that produced x.*
1. If weight$(x|_{B_1}) > 0$ then output "User 1 is guilty".
2. If weight$(x|_{B_{n-1}}) < d$ then output "User n is guilty".
3. for all $s = 2$ to $n-1$ do: Let $k =$ weight$(x|_{R_s})$. If

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

then output "User s is guilty".

One issue needs some clarification: the word x found in the illegal copy may contain some unreadable marks '?'. As a convention these bits are set to '0' before the word x is given to Algorithm 1. As a result the algorithm indeed receives a word in $\{0,1\}^l$. The correctness of Algorithm 1 is proved in the next two lemmas.

Lemma 5.2 *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 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 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.

[2]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.

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. This can be done by comparing Y to an appropriate binomial random variable.

Let X be a binomial random variable over k experiments with success probability 1/2. A routine calculation shows that for any r we have that $\Pr[Y = r] \le 2\Pr[X = r]$. This means that for any $a > 0$

$$\Pr\left[Y - \frac{k}{2} < a\right] \le 2\Pr\left[X - \frac{k}{2} < a\right] \le 2e^{-2a^2/k}$$

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

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

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 5.3 *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 5.4 *Suppose the set S is empty. Then for all s we have weight$(x|_{B_s}) \le 2s^2\log(2n/\epsilon)$.*

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:

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

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, $k' \leq 2(s+1)^2\log(2n/\epsilon)$. □

Proof of Lemma 5.3 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 5.4 shows that $\text{weight}(x|_{B_{n-1}}) \leq 2(n-1)^2\log(2n/\epsilon)$. This contradiction proves the lemma. □

This completes the proof of Theorem 5.1.

5.1 Logarithmic length c-secure codes

The n-secure code constructed in the previous section enables us to use the techniques of [8] to construct c-secure (n,l)-codes of length $l = c^{O(1)}\log n$. We thank Naor for pointing out this relation. In this section we demonstrate how to apply the simplest technique from [8] 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 [8] 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 random[3]. 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 3.2. 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 5.5 *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.

[3] In [8] the codewords of $\mathcal{C}$ are regarded as random hash functions $h : \{1,\ldots,L\} \to \{1,\ldots,n\}$.

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

1. *Apply Algorithm 1 to each of the L components of x. For each component $i = 1, \ldots, L$ 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 $y = y_1 \ldots y_L$.*

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 5.6 *Let x be a word which was produced by a coalition C of at most c users. Then with parameters as in Theorem 5.5, 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 $1 \le i \le L$ 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 $\Gamma_0(n, d)$. 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. $\square$

6 A lower bound

The following theorem provides a lower bound on the length of c-secure codes.

Theorem 6.1 *Let Γ be an (l, n) fingerprinting scheme over a binary alphabet. Suppose Γ is c-secure with ϵ error. Then the code length is at least $l \ge \frac{1}{2}(c-3)\log\frac{1}{\epsilon c}$.*

Proof Recall that $\Gamma(u, r)$ is a function mapping a user u and a random string r to a codeword in $\{0,1\}^l$. As we shall see, we may regard r as a fixed string in which case every user is assigned a unique codeword. Let $1, \ldots, c+1$ be a set of $c+1$ users. Define $B_r(k)$ to be the set of bit positions where users $1, \ldots, c+1$ see the value '1' exactly k times. More precisely:

$$B_r(k) = \{i \mid k = \#\{u \text{ s.t. } \Gamma(u,r)|_i = 1\}\}$$

Suppose $l < \frac{1}{2}(c-3)\log\frac{1}{\epsilon c}$. We show that a coalition of c users can create a word that cannot be traced. Observe that for all values of the random string r there must exist a $2 \le k_0 \le c-2$ such that

$$|B_r(k_0) \cup B_r(k_0+1)| \le \log\frac{1}{\epsilon c}$$

This follows since $\sum_{k=2}^{c-2} |B_r(k) \cup B_r(k+1)| \leq 2l < (c-3)\log\frac{1}{\epsilon c}$. Define the word W_r by

$$(W_r)_i = \begin{cases} 0 & \text{if } i \in B_r(k) \text{ for some } k \leq k_0 \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

In other words, W_r is '0' for all bit positions in blocks of weight $k \leq k_0$ and '1' in all other bit positions. We show that all coalitions of c users in $1, \ldots, c+1$ can create the word W_r with probability at least ϵ. Let C be such a coalition. To create the word W_r the coalition must first determine the value of k_0. Unfortunately, the coalition cannot do so deterministically (for a fixed k it cannot test if a certain bit position i satisfies $i \in B_r(k) \cup B_r(k+1)$ since it cannot distinguish between $i \in B_r(k+1)$ and $i \in B_r(k+2)$.) Therefore, to determine the value of k_0 the coalition simply guesses it. Since $2 \leq k_0 \leq c-2$ the correct value is guessed with probability more than $1/c$.

Let i be a detectable bit position for C. Since C includes all but one of the users in $1, \ldots, c+1$ it can determine that $i \in B_r(k) \cup B_r(k+1)$ for some k. However, it can not differentiate the case $i \in B_r(k)$ from $i \in B_r(k+1)$. Consequently, if the coalition sees the value '1' less than k_0 times in the i'th position (i.e. $\#\{u \in C \text{ s.t. } \Gamma(u,r)|_i = 1\} < k_0$) it must be case that $i \in B_r(k)$ for some $k \leq k_0$. Therefore, the coalition sets such bit positions to '0' (which is consistent with condition (1) defining W_r). Similarly if the value '1' appears at least $k_0 + 1$ times in the i'th position then $i \in B_r(k)$ for some $k \geq k_0 + 1$. Consequently, the coalition may safely set this bit position to '1'.

The case where the coalition sees '1' exactly k_0 times is a bit harder. In this case the coalition can not determine if $i \in B_r(k_0)$ or $i \in B_r(k_0+1)$. Hence it can not deterministically decide if the bit should be set to '0' or '1'. For such i the coalition flips an unbiased binary coin and sets the i'th bit to the value of the coin. The probability that all such bit positions are set correctly (i.e. the resulting word is W_r) is exactly

$$\left(\frac{1}{2}\right)^{|B_r(k_0) \cup B_r(k_0+1)|} \geq \left(\frac{1}{2}\right)^{\log\frac{1}{\epsilon c}} = \epsilon c$$

Hence, the coalition succeeds in generation W_r if two events occur simoultanously: it correctly guessed k_0 and it correctly sets the bits in locations where it sees '1' k_0 times. The probabiliy that both events simoultanously happen is at least $\epsilon c \cdot 1/c = \epsilon$. To summarize, we just proved that for any value of the random string r, any coalition C of c users in $1, \ldots, c+1$ can generate the word W_r with probability at least ϵ. It follows that the word W_r can not be traced back to a single user with probability ϵ (since no single user belongs to all these coalitions). This completes the proof of the theorem. □

7 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 rth piece which contains the rth 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 7.1 *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 [13] for the precise definitions; such systems are known to exist under some standard hardness assumptions. Let $f_k : \{0,1\}^{L'} \to \{0,1\}^{L'}$ 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

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

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 $lK = O(l^{1-1/m} L^{1/m}) = O(l L^{1/m})$.

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 $O(l/\log(L))$.

8 Conclusions

The most significant contribution of these results 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 3, we constructed codes of length $O(\log^3 n)$. For the general redistribution problem considered in Section 4 we constructed codes of length $O(\log^4 n \log^2 \frac{1}{\epsilon})$ where $\epsilon < 1/n$ is the error probability. We note that our codes are binary, hence each hidden mark need only be in one of two states. Finally, 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. The most relevant one is that of constructing shorter collusion secure codes. It seems that an n-secure code that is shorter than the one constructed in Section 5 should exist. Indeed, there is some gap between the lower bound of Section 6 and our constructed code. A shorter n-secure code will lead to an improvement in the general construction of c-secure codes as well.

Recall that throughout the paper, we assumed that secure marks can be embedded in the fingerprinted data. A mark encodes a 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 exist. Progress in this direction would be of some practical importance.

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] I. Biehl, B. Meyer, "Protocols for collusion-secure assymetric fingerprinting", Proceedings of STACS, 1997.

[4] G. Blakley, C. Meadows, G. Purdy, "Fingerprinting long forgiving messages", Proceedings of Crypto, 1985, pp. 180–189.

[5] 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.

[6] 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.

[7] Y.M. Chee, *Turán-type problems in group testing, coding theory and cryptography*, Ph.D. thesis, University of Waterloo, 1996.

[8] B. Chor, A. Fiat and M. Naor, *Tracing traitors*, Proceedings of Crypto '94, pp. 257–270.

[9] I. Cox, J. Kilian, T. Leighton, T. Shamoon, *A secure robust watermark for multimedia*, Information Hiding, LNCS 1174, Springer-Verlag, 1996, 185–206.

[10] DigiMarc Co., `http://www.digimarc.com`

[11] P. Erdös, 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.

[12] D. Glover, *The protection of computer software*, Cambridge University, 2nd ed., 1992.

[13] S. Goldwasser, *The search for provably secure cryptosystems*, AMS Lecture notes cryptology and computational number theory, 1990.

[14] M. Naor, *Private communications.*

[15] B. Pfitzmann, M. Schunter, *Assymetric fingerprinting*, Proceedings of Eurocrypt '96, pp. 84–95.

[16] B. Pfitzmann, M. Waidner, *Assymetric fingerprinting for large collusions*, Proceedings of 4th ACM Conference on Computer and Communication Security, 1997.

[17] B. Pfitzmann, M. Waidner, *Anonymous Fingerprinting*, Proceedings of Eurocrypt '97, pp. 88–102.

[18] B. Schneier, *Applied cryptography*, Wiley, 1994.

[19] D.R. Stinson, R. Wei, *Combinatorial properties and constructions of traceability schemes and frameproof codes*, SIAM J. Discrete Math., to appear.

[20] 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.

[21] van Lint, *Introduction to coding theory*, Springer-Verlag, 1982.

[22] N. Wagner, *Fingerprinting*, Proceedings of the 1983 IEEE Symposium on Security and Privacy, April, 1983, pp. 18–22.