Collusion-Secure Fingerprinting for Digital Data*

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.

*Submitted for a patent application. File #95–1199–1, Princeton University.
†Supported in part by NSF CCR–9304718.

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. 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 the effects of collusion. We discuss a general fingerprinting solution which is secure in the context of collusion. Our fingerprinting method can be applied to any form of digital data. For the remainder of this 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 [8]. 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. Goldreich and Ostrovsky discuss an oblivious RAM which prevents an opponent from gaining any information about a running program [6]. However, their scheme adds a $\log n$ slowdown and requires special hardware; we share their opinion that software can not be rigorously copy protected without the use of special hardware. Since many applications can not tolerate these restrictions, we argue for the use of fingerprints as a deterrent against software piracy. Up until now, fingerprints could not be used as undeniable proofs of guilt because of the possibility of collusion. Our results discuss fingerprints which can be traced back to their users even if multiple users collude.

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 copies.

Our fingerprint consists of a set of binary marks[1]. 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, that is it is impossible to determine if the mark is 0 or 1.

[1]For simplicity, we use binary marks throughout the paper. In Section 3.1 we show that nothing is lost by restricting the marks to be binary as opposed to n-stated.

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

There has been much research in showing that the Marking Axiom holds for a variety of domains. Wagner [11] gives a taxonomy of fingerprints and suggests subtle marks for computer software. Marks have been embedded in digital images [9], in documents [2], and in computer programs [3]. Some of these implementations refer to their markings as fingerprints. As none of these "fingerprints" is secure against collusion, we refer to them as marks. 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 (1,c)-secure. 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. An explicit construction of c-secure codes is presented in Section 4. Section 5 describes an efficient method to distribute the fingerprinted copies. 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 $I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, l\}$ we denote by $w|_I$ the word $w_{i_1} w_{i_2} \ldots w_{i_r}$ where w_i is the i'th bit of w. We refer to $w|_I$ as the *restriction* of w to the bit positions in I.

2 Protection against naive redistribution

Suppose a distributor wishes to ship its product (e.g. software, books, etc.) to a number of users. For expository purposes we assume that the product is software and that the distributor is shipping an executable. However, our results apply to other media as well.

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 an executable marked with that codeword. If we assume that users can not change the marks, then if an illegitimate copy is found, the codeword embedded in it must belong to some user u. Since, by assumption, other users could not have created an executable marked with u's codeword, user u can be held accountable for distributing the executable. 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 2.1 *An (n,l)-**code** Γ is a collection of n binary codewords $\{w^{(1)}, \ldots, w^{(n)}\} \subseteq \{0,1\}^l$. The codeword $w^{(i)}$ will be assigned to user u_i, for $1 \leq i \leq n$.*

Definition 2.2 *For a nonempty set of words* $W = \{w^{(1)}, \ldots, w^{(r)}\} \subset \{0,1\}^l$ *define*

$$[W] = \{1 \leq k \leq l \ \ s.t. \ \ w_k^{(1)} = w_k^{(2)} = \cdots = w_k^{(r)}\}$$

The set $[W]$ *is the set of bit positions on which all the words* $w^{(1)}, \ldots, w^{(r)}$ *match, i.e.* $w^{(1)}|_{[W]} = \cdots = w^{(r)}|_{[W]}$. *When* W *contains only one word define* $[W] = \{1, \ldots, l\}$.

For an (n,l)*-code* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ *and a set of users* $U = \{u_1, \ldots, u_r\} \subseteq \{1, \ldots, n\}$ *define*

$$[U;\Gamma] = [W] \quad \textit{where} \quad W = \{w^{(u_1)}, \ldots, w^{(u_r)}\}$$

Hence, $[U;\Gamma]$ *is the set of bit positions which the coalition* U *can not detect. Usually we omit the* Γ *and denote* $[U;\Gamma]$ *by* $[U]$. *We shall refer to the set of bit positions in* $[U]$ *as the* ***undetectable bits*** *of* U *and to those not in* $[U]$ *as the* ***detectable bits*** *of* U.

Definition 2.3 *Given a set of* l*-bit words* $W = \{w^{(1)}, \ldots, w^{(r)}\}$, *define the* feasible set *of* W *as*

$$F(W) = \left\{ w \in \{0,1\}^l \ \ s.t. \ \ w|_{[W]} = w^{(1)}|_{[W]} \right\}$$

Given an (n,l)*-code* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ *and a set of users* $U = \{u_1, \ldots, u_r\} \subseteq \{1, \ldots, n\}$, *define the* ***feasible set*** *of* U *as*

$$F(U;\Gamma) = F(W) \quad \textit{where} \quad W = \{w^{(u_1)}, \ldots, w^{(u_r)}\}$$

Usually we omit the Γ *and denote* $F(U;\Gamma)$ *by* $F(U)$.

Formally, the Marking Axiom discussed in the introduction states that any coalition of c users is only capable of creating an executable 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 general, the distributor would like to use a code such that, even if users collude, an unregistered copy can be traced back to some of the users who colluded. As Section 3 will show, such codes do not exist. However, we can construct codes which allow copies to be traced with high probability. These codes are introduced in Section 3.2.

In this section we consider the simpler problem of *naive redistribution*. Naive redistribution occurs when an illegitimate executable is found, and is marked by a word which belongs to one of the users. However, this need not be the case, since a group of users colluding might mark the executable with a word which does not belong to any user. The assumption of naive redistribution is still a reasonable one since naive users do not collude: they simply distribute their own copy of the executable as is.

We introduce the problem of naive redistribution through an example. Suppose a distributor ships marked copies of its product to three users: Alice, Bob, and Chuck. If an unauthorized copy is found, the distributor can examine its marks and try to determine which one of Alice, Bob or Chuck distributed the product. Suppose Alice's codeword is found in the unauthorized copy. Intuitively we would like to say that Alice is guilty of distributing the product. However, what could have happened was that Bob and Chuck colluded and were

clever enough to produce an executable marked with Alice's codeword. In this case we say that Chuck and Bob "framed" Alice. Hence, the fact that Alice's codeword is found in the executable does not necessarily prove that Alice is guilty.

We would like to construct codes that satisfy the following property: if an unauthorized copy marked by a user's codeword is found, then that user must have distributed his executable. That is, no coalition could have colluded to frame him. We usually relax this requirement and say that no coalition of at most c users could have colluded to frame the user. We call such a code a $(1, c)$-secure code. Intuitively, this notation means that the illegal copy was produced by one user, and no coalition of c users excluding u could have produced the copy. Formally we define $(1, c)$-secure codes in the following way:

Definition 2.4 *A code Γ will be called a $(1, c)$-**secure code** if every set $W \subset \Gamma$, of size at most c, satisfies $F(W) \cap \Gamma = W$.*

The definition states that in a $(1, c)$-secure 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.

Note that when a $(1, c)$-secure code is used, there is absolutely no way for a coalition to frame a single user not in the coalition[2]. As we shall see, for more general problems this is not always possible. In such cases, we must be satisfied with the condition that there is a high probability a coalition can not frame another user.

Theorem 2.1 *Let n, l, c be integers such that $l > 2^c(c + 1) \ln n$. There exists a $(1, c)$-secure (n, l)-code Γ.*

Proof We pick all n words of Γ uniformly and independently at random.
Let A be the event that Γ is not $(1, c)$-totally secure.
Let C be some coalition of c users and let y be the codeword associated with a user u not in C. Let $B(C, y)$ be the event that $y \in F(C)$. For the event $B(C, y)$ to occur, for all l bit positions, y may not contain a 0 when all users in C see a 1 or vice versa. Hence,

$$\Pr[B(C, y)] = \left[1 - \left(\frac{1}{2}\right)^c\right]^l$$

The number of different events $B(C, y)$ where the coalition C contains at most c users and $y \notin C$ is

$$\sum_{k=1}^{c} \binom{n}{k}(n - k) \leq n^{c+1}$$

For the event A to occur, one of the events $B(C, y)$ must occur. Hence,

$$\Pr[A] \leq \sum_{C,y} \Pr[B(C, y)] \leq n^{c+1}\left[1 - \left(\frac{1}{2}\right)^c\right]^l$$

[2]Under the assumption that coalitions can only generate words within their feasible set.

When $\Pr[A] < 1$ the required code must exist. Hence, we require that

$$n^{c+1}\left[1 - \left(\frac{1}{2}\right)^{c}\right]^{l} < 1$$

This inequality is satisfied when

$$l > 2^{c}(c+1)\ln n$$

□

The bound proved in Theorem 2.1 is very weak, but for small c it is still useful. The code constructed in the theorem is a random code, meaning the codewords are chosen at random. 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 $(1, c)$-codes

We will now show a deterministic construction for $(1, c)$-totally secure 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 $(1, n)$-totally secure (n, n)-code. For $i = 1, \ldots, n$, let $z^{(i)}$ be the l-bit word whose i'th bit is 1 and the rest are 0. Define the code $\Gamma_0(n)$ to be

$$\Gamma_0(n) = \{z^{(1)}, \ldots, z^{(n)}\}$$

For example, the code $\Gamma_0(5)$ for the five users A, B, C, D, E is as follows:

A	10000
B	01000
C	00100
D	00010
E	00001

Claim 2.2 *$\Gamma_0(n)$ is a $(1, n)$-totally secure (n, n)-code.*

Proof Let $W \subseteq \Gamma_0$ be a coalition. Assume $W = \{z^{(i_1)}, \ldots, z^{(i_r)}\}$. The detectable bit positions of W are $i_1, \ldots, i_r$. Let i be an undetectable bit position, then the i'th bit of all the words in W must be 0. Hence, the feasible set of W will contain any word x such that $x|_{[W]}$ is all zeroes. The only words in $\Gamma_0(n)$ satisfying this property are the words in W. Thus we find that $F(W) \cap \Gamma_0(n) = W$, which proves that the code is $(1, n)$-secure. □

Note that any $(1, n)$-secure code must have length at least n. This follows because 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.

We now use the code Γ_0 to construct larger codes. The construction will be recursive in nature, and the code Γ_0 will serve as the base case. We recall some basic definitions from the theory of error correcting codes. See [10] for more details.

Definition 2.5 *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 use an error correcting code to build a large code (a code for many users) from a smaller one. Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ be a $(1, c)$-secure (p, l)-code and let $\mathcal{C}$ be a $(L, N, D)_p$-ECC. We construct a $(1, c)$-secure (N, lL)-code Γ' in the following way: Let $v = v_1 v_2 \ldots v_L$ be a codeword in $\mathcal{C}$ where the v_i's are the letters comprising v (i.e., each v_i is a number between 1 and p). Define the word W_v in the following way:

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

Where $\|$ means concatenation of strings. The code Γ' is defined as

$$\Gamma' = \{W_v \mid v \in \mathcal{C}\}$$

Clearly, Γ' is a (N, lL)-code. We show that it is $(1, c)$-secure.

Lemma 2.3 *The code Γ' is a $(1, c)$-secure 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 corresponding to 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 $v \in \mathcal{C}$ be the codeword from which W was derived. For all $k = 1, \ldots, c$, the words v and v_k match in less than L/c positions. This follows since the minimal Hamming distance of $\mathcal{C}$ is bigger than $L(1 - 1/c)$. It follows that the codewords $v^1, \ldots, v^k$ match v in less than L positions. Hence, there must exist a position $1 \leq j \leq L$ for which $v_j \neq v_j^k$ for all $k = 1, \ldots, c$.

Since Γ is a $(1, c)$-secure code we know that

$$w^{(v_j)} \notin F(\{w^{(v_j^1)}, \ldots, w^{(v_j^c)}\}; \Gamma)$$

Since $w^{(v_j)}$ is a subword of W, this implies that $W \notin F(C; \Gamma')$. This contradiction proves the lemma. □

We summarize the construction in the following theorem:

Theorem 2.4 *Let Γ be a $(1, c)$-secure (p, l)-code and let $\mathcal{C}$ be a $(L, N, D)_p$-ECC where $D > L(1 - 1/c)$. Then there exists an explicit $(1, c)$-secure (N, lL)-code.*

Note that the error correcting codes we are using have very large minimal distance. Many of the known constructions in coding theory do not yield good codes for such large minimal distances[3]. We now describe a very simple ECC which can be effectively used to construct good $(1, c)$-secure codes.

Lemma 2.5 *There exists a $(p, p^{d+1}, p-d)_p$ ECC, denoted by $\mathcal{C}(p, d)$, for any prime p and $d < p$.*

Proof The code is very simple. Consider all polynomials of degree d over $\mathbb{F}_p$. To every polynomial $f \in \mathbb{F}_p[x]$ we associate the codeword w_f, defined as follows:

$$w_f = f(0)f(1)\dots f(p-1)$$

The code $\mathcal{C}(p, d)$ is defined as

$$\mathcal{C}(p, d) = \{w_f \mid f \in \mathbb{F}_p[x] \text{ and } \deg(f) \leq d\}$$

Observe that two polynomials of degree d over $\mathbb{F}_p$ can intersect at no more than d points. This follows since their difference is a polynomial of degree at most d and hence has at most d zeroes. Thus, two codewords can have at most d common letters, implying that the minimal distance of $\mathcal{C}(p, d)$ is $p - d$. □

Since the sum of two polynomials is a polynomial, the code $\mathcal{C}(p, d)$ is a linear code. Furthermore, it satisfies the singleton bound [10, p. 57] and hence is an MDS code. This means that the code is optimal in the sense that there is no code with the same length and minimal distance which contains more codewords.

To construct an (n, l)-code Γ which is $(1, c)$-secure, we start from the code $\Gamma_0(c)$ and iteratively construct larger and larger codes using the codes $\mathcal{C}(p, d)$ and Theorem 2.4. For simplicity, assume c is prime. This is not a restriction since we can always increase c to the closest prime.

We analyze the results of running once through this iterative process. To insure the final code contains n codewords, we must choose a p such that $p^{d+1} \geq n$, since this is the number of codewords in the $\mathcal{C}(p, d)$ code. To satisfy the condition of Theorem 2.4 we get the additional constraint that $p - d > p - p/c$, that is, $p > cd$. Choosing $p = c\log(n)$ and $d = \lfloor \log(n) \rfloor$ satisfies the two constraints (we make no attempt to optimize the expressions). Combining the codes $\Gamma_0(p), \mathcal{C}(p, d)$ and Theorem 2.4, we get a $(n, c^2 \log^2(n))$-code which is $(1, c)$-secure. We summarize this result in the following corollary.

Corollary 2.6 *For any n, c, there are explicit $(n, c^2 \log^2(n))$-codes which are $(1, c)$-secure.*

This bound should be contrasted with the one given in Theorem 2.1. Iterating this process several times will reduce the dependence on $\log n$ at the expense of increasing the dependence on c.

[3]The Reed-Solomon codes yield good codes in these ranges. However, our $\mathcal{C}(p, d)$ code, which is described in Lemma 2.5, is simpler and just as good.

3 Definition of 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 executable which is marked by a word which is not a codeword. When such an executable is found, $(1, c)$-codes are not guaranteed to provide any information as to who generated the executable.

3.1 Totally secure codes

In general, users can collude to create unregistered executables. Assume a distributor marks an executable with a code Γ. Now, suppose a coalition of users, C, colludes to generate an unregistered executable marked by the word x and then distributes this new executable. When this unregistered executable is found, the distributor would like to detect a subset of the coalition C who distributed it.

We illustrate the problem with an example. Let Γ be some code. Suppose C_1 and C_2 are two coalitions of c users each such that $C_1 \cap C_2 = \emptyset$. Assume that the feasible sets of the two coalition have a nonempty intersection, that is $F(C_1) \cap F(C_2) \neq \emptyset$. If an unregistered executable is found which is marked by a codeword $x \in F(C_1) \cap F(C_2)$, then both C_1 and C_2 are suspect. Since their intersection is empty, we can not determine with certainty who created the unregistered executable. 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 3.1 *A code Γ will be called a **totally c-secure** if*

$$U_1 \cap \ldots \cap U_r = \emptyset \quad \Rightarrow \quad F(U_1) \cap \ldots \cap F(U_r) = \emptyset$$

For all coalitions $U_1, \ldots, U_r$ of at most c users each.

Let Γ be a totally c-secure code which we use to mark some executable E that we wish to distribute. Let E' be a copy of E which is marked by the word x. Then, under the assumption that at most c users can collude, the following algorithm will output a set of users who must have been a part of the coalition that created the executable E'.

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

1. Find all sets of at most c users $U_1, \ldots, U_r$, where $x \in F(U_i)$.
2. Calculate $U = \bigcap_{i=1}^{r} U_i$.

It follows immediately from the definition of totally secure codes that Algorithm 1 will always output a nonempty set of users who created the unregistered executable E'. We summarize this in the following proposition.

Proposition 3.1 *Let Γ be a totally c-secure code. If a coalition U of at most c users generated an executable marked with the word x, then Algorithm 1 will output a nonempty subset of U.*

Proof Since $x \in \cap_{i=1}^{r} F(U_i)$, we know that $\cap_{i=1}^{r} F(U_i)$ is not empty. Therefore, because Γ is totally c-secure, $U = \cap_{i=1}^{r} U_i$ must be nonempty. It is clear that one of the coalitions $U_1, \ldots, U_r$ must have generated the word x, and therefore U must be a subset of the actual coalition that generated x. □

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. This is proved in the following theorem.

Theorem 3.2 *For $c \geq 2$ and $n \geq 3$ there are no totally c-secure (n, l)-codes.*

Proof Clearly, it is enough to show that there are no totally 2-secure codes. Let Γ be an arbitrary (n, l)-code. We show that any set of three users u_1, u_2, u_3 satisfies:

$$A \stackrel{def}{=} F(\{u_1, u_2\}) \cap F(\{u_2, u_3\}) \cap F(\{u_1, u_3\}) \neq \emptyset$$

This will prove that Γ is not 2-secure, since the intersection of the three pairs $\{u_1, u_2\}$, $\{u_2, u_3\}$, $\{u_1, u_3\}$ is empty.

Let u_1, u_2, u_3 be three distinct, arbitrary users and let $c^{(1)}, c^{(2)}, c^{(3)} \in \Gamma$ be their corresponding codewords in Γ. Define the word $w \in \{0,1\}^l$ as follows: the i'th bit of w is defined by

$$w_i = \mathrm{MAJ}(c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$$

where MAJ is the majority function. Recall that the codewords $c^{(1)}, c^{(2)}, c^{(3)}$ are binary strings and therefore the majority function is well defined.

We show that $w \in A$, implying that A is not empty. By symmetry, to show that $w \in A$, it suffices to show that $w \in F(\{u_1, u_2\})$. The words $c^{(1)}$ and $c^{(2)}$ either match or mismatch in their i'th bit. If $c^{(1)}$ and $c^{(2)}$ match on the i'th bit then, by definition of w, we have $w_i = c_i^{(1)}$. Otherwise, u_1, u_2 can detect the i'th bit position which means they can set their i'th bit to w_i. Hence, $w \in F(\{u_1, u_2\})$. □

Note that the above proof relies heavily on the fact that the code Γ is made up of binary strings. Without this assumption, we can not use the majority function. One could hope that by using strings over a larger alphabet, totally c-secure codes for $c > 1$ could be constructed; however, this is not the case. To see this, we must first understand how to extend the definition of total security to codes over an alphabet of arbitrary size. A subtlety arises in the definition of the feasible sets.

An $(n, l)_p$-code Γ is a set of n codewords, each of length l, over an alphabet of size p. The detectable and undetectable bits are defined analogously to the binary case. We consider three ways of defining the feasible set of a coalition U.

Definition 3.2 *Let* $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ *be an* $(n, l)_p$*-code. Let* $U = \{u_1, \ldots, u_c\}$ *be some coalition. Define*

$$\begin{aligned} F_p(U) &= \left\{ x \in \{0, \ldots, p-1\}^l \mid x|_{[U]} = w^{(u_1)}|_{[U]} \right\} \\ \tilde{F}_p(U) &= \left\{ x \in \{0, \ldots, p-1\}^l \mid \forall i = 1, \ldots, l : x_i \in \{w_i^{(u_1)}, w_i^{(u_2)}, \ldots, w_i^{(u_c)}, ?\} \right\} \\ \hat{F}_p(U) &= \left\{ x \in \{0, \ldots, p-1\}^l \mid \forall i = 1, \ldots, l : x_i \in \{w_i^{(u_1)}, w_i^{(u_2)}, \ldots, w_i^{(u_c)}\} \right\} \end{aligned}$$

The definition of $F_p(U)$ states that whenever a coalition can detect a mark, they can change it to any state. The definition of $\tilde{F}_p(U)$ is more stringent and says that whenever a coalition detects a mark, it can only change it to one of the states it sees or make it unreadable (denoted by '?'). A mark is unreadable if the distributor who embedded the mark in the executable can not determine its state. Finally, the definition of $\hat{F}_p(U)$ is the same as $\tilde{F}_p(U)$ except the coalition is unable to make the mark unreadable.

Clearly, the definition of $F_p(U)$ and $\tilde{F}_p(U)$ are reasonable in terms of what we allow a coalition to do. The definition of $\hat{F}_p(U)$ is, in most domains, too weak since users can usually make a detectable mark unreadable. We now extend Definition 3.1 and define total security for a code over an arbitrary alphabet. We get three different definitions using $F_p, \tilde{F}_p$ and $\hat{F}_p$.

Definition 3.3 *An* $(n, l)_p$*-code* Γ *will be called* $F, \tilde{F}, \hat{F}$ totally c-secure *if*

$$\begin{aligned} U_1 \cap \ldots \cap U_r = \emptyset \quad &\Rightarrow \quad F_p(U_1) \cap \ldots \cap F_p(U_r) = \emptyset \\ U_1 \cap \ldots \cap U_r = \emptyset \quad &\Rightarrow \quad \tilde{F}_p(U_1) \cap \ldots \cap \tilde{F}_p(U_r) = \emptyset \\ U_1 \cap \ldots \cap U_r = \emptyset \quad &\Rightarrow \quad \hat{F}_p(U_1) \cap \ldots \cap \hat{F}_p(U_r) = \emptyset \end{aligned}$$

respectively, for all coalitions $U_1, \ldots, U_r$ *of at most* c *users each.*

The proof of Theorem 3.2 can be extended to show that F and $\tilde{F}$ totally secure codes do not exist. However, $\hat{F}$ totally secure codes do exist. Such codes will be constructed and used in Section 4.2.

3.2 Probabilistically secure codes

The negative result proved in Theorem 3.2 forces us to weaken our requirements for marking schemes. We can not require that there be no way for a coalition to pretend to be a different coalition. Instead, we require that it be a low probability event. To understand what the probability space is, we must first digress and make a simple observation regarding how codewords are embedded in the executable we are trying to protect.

Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ be an (n, l)-code. Suppose we wish to protect an executable E using the code Γ. User u_i will receive a copy of E marked by the word $w^{(i)}$. The order in which the bits of $w^{(i)}$ are embedded in the executable is arbitrary and is not known to the users. Before we begin producing copies of E marked with codewords, we pick a random

permutation[4] $\pi \in S_l$. We now mark u_i's copy of the executable E with the codeword $\pi w^{(i)}$. The same permutation π must be used for all users, that is for all $i = 1, \ldots, n$, user u_i sees the word $\pi w^{(i)}$. The permutation π is kept secret from the users.

Other methods similar to permuting the codewords are also possible. For now we restrict our attention to permutations since it makes the definitions much simpler, and is sufficient for the construction of good c-secure codes. We next analyze what strategies can be employed by a coalition of users when producing an unregistered copy of E.

Let C be a coalition of c users. Before we give the formal definition of a strategy, we give some intuition as to what the coalition can do. The coalition can detect some of the bit positions used for encoding the codewords in E. Let d be the number of bit positions the coalition can detect. The users in C can now apply an algorithm, which might use some random bits, to change the d bits they detect. The result will be an executable E' marked by some word $x \in \{0,1\}^l$. The word x depends on the algorithm used by the coalition, the random string the algorithm uses, and the permutation $\pi \in S_l$ used in embedding the codewords into the executable. The dependence on π stems from the fact that for different π, the coalition sees different bit patterns and the algorithm will output different values. Note that once the random bits are chosen, the algorithm used by the coalition can be treated as a function which is given cd bits (each of the c users detects d bits) and outputs d bits.

The following definitions will be useful in describing c-secure codes. Consider the following subgroup of permutations:

Definition 3.4 *For a coalition C, denote by SYM(C) the set of permutations σ in S_l which satisfy the following two conditions:*

1. *σ fixes pointwise the undetectable bits of C, i.e.*

$$\forall\, i \in [C] \quad : \quad \pi(i) = i$$

2. *For every user $u \in C$, σ leaves the word $w^{(u)}$ unchanged, i.e.*

$$\forall\, u \in C \quad : \quad \pi w^{(u)} = w^{(u)}$$

It is easily verified that SYM(C) is a subgroup of S_l. For a word $x \in \{0,1\}^l$, we denote by SYM(x, C) the set of words

$$\mathrm{SYM}(x, C) = \{\ \pi x \ \mid\ \pi \in \mathrm{SYM}(C)\}$$

The sets SYM(x, C) will be referred to as **C-symmetry sets.**

The importance of the group SYM(C) is that for any $\pi \in S_l$ and $\sigma \in \mathrm{SYM}(C)$, the coalition C sees exactly the same bit pattern whether π or $\pi\sigma$ is used. This follows immediately from condition 2. Hence, for both these permutations, the coalition will generate the same changed

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

marks. Let $\pi^{-1}x \in \{0,1\}^l$ be the word the coalition generates when π is used. When $\pi\sigma$ is used, the coalition generates the word $\sigma^{-1}\pi^{-1}x$. Simplifying this, we see that if the coalition generates the word $y \in \{0,1\}^l$ when π is used, it will generate the word $\sigma^{-1}y$ when $\pi\sigma$ is used.

Example We demonstrate this phenomenon using the following (3,6)-code.

	1	2	3	4	5	6
A	0	0	0	0	0	1
B	0	1	1	1	1	1
C	1	0	0	1	1	0

The top numbers indicate the bit positions. Consider the coalition AB. Its feasible set is

$$F(AB) = 0 * * * *1$$

where $*$ means either 0 or 1. The symmetry group of AB consists of all permutations that permute columns 2, 3, 4, 5 and fix columns 1 and 6. Suppose we take π to be the identity permutation and σ to be a cyclic shift of 2-5, i.e. the permutation $(5,2,3,4)$. We will observe the relation between the words generated by AB when π is used and the words generated when $\pi\sigma$ is used. In the two cases, the coalition sees the following words:

π

	1	2	3	4	5	6
A	0	0	0	0	0	1
B	0	1	1	1	1	1

$\pi\sigma$

	1	5	2	3	4	6
A	0	0	0	0	0	1
B	0	1	1	1	1	1

Since the coalition observes the same bit pattern in both cases, they will produce the same output. Suppose the coalition generates the word 000111, which is in its feasible set. Then the resulting executables will be marked in the following way:

π

1	2	3	4	5	6
0	0	0	1	1	1

$\pi\sigma$

1	5	2	3	4	6
0	0	0	1	1	1

When we read the word encoded in the executable, we find the following words:

π

1	2	3	4	5	6
0	0	0	1	1	1

$\pi\sigma$

1	2	3	4	5	6
0	0	1	1	0	1

This example demonstrates that if users generate the word x when π is used to encode the marks, then they generate the word $\sigma^{-1}x$ when $\pi\sigma$ is used. We can now state the formal definition of a strategy for a coalition.

Definition 3.5 *Let Γ be an (n,l)-code and C a coalition of c users. A **strategy** for C is a function*

$$f : S_l \times \{0,1\}^* \to \{0,1\}^l$$

such that

1. *for all π and random bits R we have $f(\pi, R) \in F(C)$, and*

2. for all $\sigma \in SYM(C)$: $\quad f(\pi\sigma, R) = \sigma^{-1} f(\pi, R)$.

Thus, a strategy for a coalition C must produce words within C's feasible set. Furthermore, if π is the permutation used to embed the code in the executable, then the strategy can only depend on which coset π is in. Therefore, if π and π' are in the same coset, then the words generated by the coalition must be related as in condition 2 above.

Next, we want to say that a set of words can be specified a-priori such that no matter what strategy the coalition uses, they will not be able to generate words from the set, with high probability. Such a set will be called a hard set. The complement of a hard set is a set such that no matter which strategy is used, the word produced by the coalition will be in the set, with high probability. Such a set will be called an achievable set. Formally, we define hard sets and achievable sets as follows:

Definition 3.6 *A set* $A \subseteq F(C)$ *is* ϵ***-achievable*** *for* C *if every strategy* f *for* C *satisfies*

$$\Pr_{\pi,R}[f(\pi, R) \in A] > 1 - \epsilon$$

A set $H \subseteq F(C)$ *is* ϵ***-hard*** *for* C *if every strategy* f *for* C *satisfies*

$$\Pr_{\pi,R}[f(\pi, R) \in A] < \epsilon$$

Observe that H is ϵ-hard for C if and only if $F(C) - H$ is an ϵ-achievable set for C. With these definitions in mind, we are ready to define c-secure codes.

Definition 3.7 *A code* Γ *is* c***-secure with*** ϵ***-error*** *if for every coalition* C*, it is possible to assign a set* A_C *which is* ϵ*-achievable for* C *such that*

$$(1) \qquad C_1 \cap \ldots \cap C_r = \emptyset \quad \Rightarrow \quad A_{C_1} \cap \ldots \cap A_{C_r} = \emptyset$$

For all sets $C_1, \ldots, C_r$ *of at most* c *users each.*

Notice the similarity to the definition of totally secure codes (Definition 3.1). The only difference is that the feasible sets are replaced by achievable sets, which are smaller. We refer to condition 1 as the *security condition.* Algorithm 1 now becomes:

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

1. Find all sets of at most c users $C_1, \ldots, C_r$ where $x \in A_{C_i}$.
2. If $r = 0$, i.e. x is not in any A_C, output "Failure" and stop.
3. Calculate $C = \bigcap_{i=1}^{r} C_i$.

Notice that the selection of achievable sets is done before Algorithm 2 is applied to the word x. This a priori selection of the achievable sets is crucial.

Proposition 3.3 *Let Γ be a c-secure code with ϵ-error. If a coalition C of at most c users generated an executable marked with a word x, then Algorithm 2 will output a nonempty subset of C with probability at least $1 - \epsilon$.*

Proof Let f be some strategy for the coalition C. Let $S \subseteq \{0,1\}^l$ be the set of words such for any $x \in S$ Algorithm 2 outputs a subset of C. We want to show that

$$\Pr_{\pi,R}[f(\pi, R) \in S] > 1 - \epsilon$$

Clearly, if $f(\pi, R) \in A_C$ then $f(\pi, R) \in S$, since in this case the algorithm would necessarily output a subset of C. Hence,

$$\Pr_{\pi,R}[f(\pi, R) \in S] \geq \Pr_{\pi,R}[f(\pi, R) \in A_C] > 1 - \epsilon$$

□

Algorithm 2 has the potential of failing to output any user which can be blamed. This occurs in step 2 of the algorithm. Proposition 3.3 shows that this is a low probability event.

3.3 Structure of achievable sets

We now turn our attention to providing tools for proving that codes are c-secure. The definition of achievability is hard to work with. It requires detailed analysis of the possible strategies coalitions can use. Fortunately, there is a simpler equivalent characterization which is provided in the following theorem.

Theorem 3.4 *Let C be a coalition. A set A is ϵ-achievable for C if and only if for every $y \in F(C)$*

$$\Pr_x[x \in A \mid x \in SYM(y, C)] \stackrel{def}{=} \frac{|\{A \cap SYM(y, C)\}|}{|SYM(y, C)|} > 1 - \epsilon \tag{2}$$

Proof Assume condition (2) holds for A. We show that A is ϵ-achievable. Suppose A is not ϵ-achievable; then there exists a strategy $f(\pi, R)$ such that

$$\Pr_{\pi,R}[f(\pi, R) \in A] \leq 1 - \epsilon$$

There must exist an R_0 and a coset Σ of $\mathrm{SYM}(C)$ for which

$$\Pr_{\pi \in \Sigma}[f(\pi, R_0) \in A] \leq 1 - \epsilon$$

otherwise the above condition will not be satisfied for the entire space. Pick some $\pi_0 \in \Sigma$ and let $y = f(\pi_0, R_0)$. By definition, we know that $y \in F(C)$ and for any $\sigma \in \mathrm{SYM}(C)$

$$f(\pi_0 \sigma, R_0) = (\sigma^{-1})y$$

As σ varies over $\mathrm{SYM}(C)$, $(\sigma^{-1})y$ will vary over all of $\mathrm{SYM}(y, C)$. Hence,

$$\begin{aligned}\Pr_x[x \in A \quad | \quad x \in \mathrm{SYM}(y,C)] &= \Pr_{\sigma \in \mathrm{SYM}(C)}[(\sigma^{-1})y \in A] \\ &= \Pr_{\sigma \in \mathrm{SYM}(C)}[f(\pi_0\sigma, R_0) \in A] = \Pr_{\pi \in \Sigma}[f(\pi, R_0) \in A] \leq 1 - \epsilon\end{aligned}$$

This proves that condition (2) is violated in the C-symmetry set $\mathrm{SYM}(y, C)$. This contradiction shows that A is ϵ-achievable.

Conversely, suppose A is ϵ-achievable for C. We show that A satisfies condition (2). Let $\pi_1, \ldots, \pi_t \in S_l$ be arbitrary coset representatives for $\mathrm{SYM}(C)$, one for each coset of $\mathrm{SYM}(C)$. Any permutation $\pi \in S_l$ can be uniquely written as $\pi = \pi_j\sigma$ for some j and $\sigma \in \mathrm{SYM}(C)$.

For $y \in F(C)$ define the following strategy $f_y(\pi, R)$ for C: Write $\pi = \pi_j\sigma$ where $\sigma \in \mathrm{SYM}(C)$ and define

$$f_y(\pi, R) = (\sigma^{-1})y$$

This strategy is well defined, since any $\pi \in S_l$ can be uniquely written as $\pi = \pi_j\sigma$. Note that the strategy ignores the random string R.

First we show that f_y is a valid strategy for C. When $y \in F(C)$ and $\sigma \in \mathrm{SYM}(C)$ it follows from the definitions that $\sigma^{-1}y \in F(C)$. Hence, for all π, R the strategy f_y satisfies $f_y(\pi, R) \in F(C)$. Next, for any $\tau \in \mathrm{SYM}(C)$ and $\pi = \pi_j\sigma \in S_l$ we have

$$f_y(\pi\tau, R) = f_y(\pi_j\sigma\tau, R) = (\tau^{-1})(\sigma^{-1})y = (\tau^{-1})f_y(\pi_j\sigma, R) = (\tau^{-1})f_y(\pi, R)$$

Hence, $f_y(\pi, R)$ is a valid strategy for C. Since A is ϵ-achievable for C, we know that

$$\Pr_{\pi,R}[f_y(\pi, R) \in A] > 1 - \epsilon$$

As π varies over the different permutations in $\mathrm{SYM}(C)$, $f_y(\pi, R)$ will vary over the entire set $\mathrm{SYM}(y, C)$. The same will be true for any R and any coset of $\mathrm{SYM}(C)$. Hence,

$$\begin{aligned}\Pr_x[x \in A \quad | \quad x \in \mathrm{SYM}(y,C)] &= \Pr_{\pi \in \mathrm{SYM}(C)}[(\pi^{-1})y \in A] \\ &= \Pr_{\pi \in \mathrm{SYM}(C)}[f_y(\pi, R) \in A] = \Pr_{\pi,R}[f_y(\pi, R) \in A] > 1 - \epsilon\end{aligned}$$

Since the choice of y was arbitrary, this holds for any $y \in F(C)$. □

4 Construction of c-secure codes

The idea of constructing c-secure codes is similar to the method used in Section 2.1. There, we began by presenting an (n, l)-code which is $(1, n)$-secure and then showed how to construct larger codes from it[5]. In this section we follow a similar scheme. We first construct an (n, l)-code which is n-secure. 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 whose length is $\log^{O(1)}(n)$ where $c = O(\log n)$.

[5]By a larger code, we mean an (n', l')-code which is $(1, n)$-secure where $n' > n$.

4.1 An n-Secure Code

In this section, we present an (n, l)-code which is n-secure with ϵ-error for any $\epsilon > 0$. The code is simple to describe. For $n = 5$ it looks like this:

A	111111111111
B	000111111111
C	000000111111
D	000000000111
E	000000000000

Formally, 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 $c_1, \ldots, c_{m-1}$, each duplicated d times. The code in the example is $\Gamma_0(5, 3)$. The amount of duplication will determine the error probability ϵ. The rest of this section is devoted to proving the following theorem:

Theorem 4.1 *For $n \geq 3$ and any $d > 1$, the code $\Gamma_0(n, 2dn^2 \log n)$ is n-secure with error $2/n^{d-1}$.*

For example, choosing $d = 2$ will produce a code for n users whose length is $4(n-1)n^2 \log n = O(n^3 \log n)$ which is n-secure with error $2/n$.

The crux of the proof lies in proving that the code is $(n-1)$-secure. Any code which is $(n-1)$-secure with ϵ-error is trivially n-secure with ϵ-error. This follows by defining the achievable set of the coalition of all n users to be their entire feasible set. It follows that with this assignment, the code becomes n-secure with ϵ-error. Hence, from here on, we will only be concerned with proving that the code is $(n-1)$-secure.

Recall that the definition of a c-secure code requires that we assign an achievable set to every coalition of at most c users. Intuitively, the difficult part is assigning achievable sets to coalitions of exactly c users. Thus, once we assign achievable sets to all coalitions of exactly c users, we would like to have a general method for assigning achievable sets to the smaller coalitions. This is provided by the following theorem:

Theorem 4.2 *Let Γ be an (n, l)-code and let $c < n$. Suppose we assign a set A_C which is ϵ-achievable to each coalition C of exactly c users, where for all sets of coalitions $C_1, \ldots, C_r$ of exactly c users*

$$C_1 \cap \ldots \cap C_r = \emptyset \quad \Rightarrow \quad A_{C_1} \cap \ldots \cap A_{C_r} = \emptyset$$

Assuming $\epsilon < 1/c$, it is possible to assign achievable sets to all coalitions of at most c users, so that Γ becomes c-secure with $c\epsilon$-error[6]*.*

Proof Let C be a coalition of at most c users. We first show how to define the achievable set of C. Since $c < n$, it is always possible to find c coalitions $U_1, \ldots, U_c$ of *exactly* c users each, such that

$$U_1 \cap \ldots \cap U_c = C \tag{3}$$

[6]When $c < n/2$, the error degrades to 2ϵ as opposed to $c\epsilon$ as stated in the theorem. This will become apparent in the proof. For simplicity, we ignore this improvement.

In fact, when c is small, this can be done with less than c coalitions. For example, when $c < n/2$, two coalitions suffice. However, when $c = n - 1$ and C contains one user, we must use $n-1$ coalitions to satisfy condition (3). This follows from the fact that the intersection of k distinct sets of size $n-1$ contains $n-k$ elements. For simplicity, we assume that c coalitions are always required to satisfy condition (3).

Define the achievable set of C as

$$A_C = F(C) \cap \bigcap_{i=1}^{c} A_{U_i}$$

To prove the theorem, we must show that A_C is ϵc-achievable for C, and that this assignment satisfies the security condition for all coalitions of at most c users. To show that A_C is ϵc-achievable for C, we use Theorem 3.4. Let $y \in F(C)$. We show that

$$\Pr_x[x \in A_C \mid x \in \mathrm{SYM}(y, C)] > 1 - \epsilon c$$

Since $C \subseteq U_i$, for every $x \in \mathrm{SYM}(y, C)$ we have that $\mathrm{SYM}(x, U_i) \subseteq \mathrm{SYM}(y, C)$. This follows since $\mathrm{SYM}(U_i)$ is a subgroup of $\mathrm{SYM}(C)$. It follows that $\mathrm{SYM}(y, C)$ can be partitioned into U_i-symmetry sets

$$\mathrm{SYM}(y, C) = \mathrm{SYM}(y_1, U_i) \cup \mathrm{SYM}(y_2, U_i) \cup \ldots \cup \mathrm{SYM}(y_r, U_i)$$

For some $y_1, \ldots, y_r \in \mathrm{SYM}(y, C)$. This is indeed a partition since different U_i-symmetry sets are disjoint. Since A_{U_i} is ϵ-achievable for U_i it must contain at least a $1 - \epsilon$ fraction of $\mathrm{SYM}(y_j, U_i)$ for all $j = 1, \ldots, r$. Hence, A_{U_i} must contain at least a $1 - \epsilon$ fraction of $\mathrm{SYM}(y, C)$. Combining this with fact that $\mathrm{SYM}(y, C) \subseteq F(C)$ we get

$$\Pr_x[x \in A_{U_i} \cap F(C) \mid x \in \mathrm{SYM}(y, C)] = \Pr_x[x \in A_{U_i} \mid x \in \mathrm{SYM}(y, C)] > 1 - \epsilon$$

Since y was arbitrary, the above statement holds for all $y \in F(C)$. Hence, $A_{U_i} \cap F(C)$ is ϵ-achievable for C. From this it follows that A_C is ϵ-achievable for C in the following way:

$$\begin{aligned}
\Pr_x[x \notin A_C \mid x \in \mathrm{SYM}(y, C)] &= \Pr_x\left[x \notin \bigcap_{i=1}^{c-1} A_{U_i} \cap F(C) \mid x \in \mathrm{SYM}(y, C)\right] \\
&\leq \sum_{i=1}^{c-1} \Pr_x\left[x \notin A_{U_i} \cap F(C) \mid x \in \mathrm{SYM}(y, C)\right] < (c-1)\epsilon < c\epsilon
\end{aligned}$$

This proves that

$$\Pr_x[x \in A_C \mid x \in \mathrm{SYM}(y, C)] > 1 - c\epsilon$$

for all $y \in F(C)$. Hence, A_C is $c\epsilon$-achievable for U.

To conclude the proof of the theorem, we have to show that this assignment of achievable sets to small coalitions satisfies the security condition. Let $C_1, \ldots, C_k$ be a collection of coalitions, each with at most c users, such that $C_1 \cap \ldots \cap C_k = \emptyset$. We want to show that $A_{C_1} \cap \ldots \cap A_{C_k} = \emptyset$.

Let $U_i^1, \ldots, U_i^c$ be the set of coalitions, each containing exactly c users, that were used to define the achievable set A_{C_i}. We know that

$$\bigcap_{i=1}^{k} \bigcap_{j=1}^{c} U_i^j = \bigcap_{i=1}^{k} C_i = \emptyset$$

Hence, by the assumption of the theorem

$$\bigcap_{i=1}^{k} A_{C_i} = \bigcap_{i=1}^{k} \bigcap_{j=1}^{c} A_{U_i^j} = \emptyset$$

□

Theorem 4.2 shows that to prove the code $\Gamma_0(n, d)$ is $(n-1)$-secure, it suffices to assign achievable sets to coalitions of exactly $n-1$ users and show that the security condition holds for this assignment. Before we assign achievable sets, we must introduce some notation. Consider the code $\Gamma_0(n, d)$.

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. Let U_s be the coalition of $n-1$ users excluding user s.

3. Let R_s be the set of bit positions in which exactly $s-1$ users in U_s see a 1. Thus, for $1 < s < n$, we have $R_s = B_{s-1} \cup B_s$.

For a string $x \in \{0,1\}^l$, let weight(x) denote the number of 1's in x. Consider the code $\Gamma_0(n, 2dn^2 \log n)$. For this code, we assign achievable sets to coalitions U_s of exactly $n-1$ users in the following way:

1. for $s = 1, n$, define $A_{U_s} = F(U_s)$.

2. for $1 < s < n$, define A_{U_s} to be the set of all $x \in F(U_s)$ such that if $k = \text{weight}(x|_{R_s})$ then

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

Intuitively, this definition says that x is achievable for U_s if the 1's in $x|_{R_s}$ are roughly evenly distributed between B_{s-1} and B_s. It remains to show that the sets A_{U_s} are indeed achievable and that they satisfy the assumptions of Theorem 4.2. These two properties are proved in the next two lemmas.

Lemma 4.3 *Consider the code $\Gamma_0(n, 2dn^2 \log n)$ and define the achievable sets A_{U_s} as above. Then, for all $s = 1, \ldots, n$, the set A_{U_s} is $2/n^d$-achievable for U_s.*

Proof For $s = 1, n$, the proof is trivial. For $s = 2, \ldots, n-1$, let $y \in F(U_s)$ and set $k = \text{weight}(y|_{R_s})$. We show that

$$\Pr_x[x \in A_{U_s} \mid x \in \text{SYM}(y, U_s)] \geq 1 - \frac{2}{n^d}$$

It suffices to show that

$$\Pr_x \left[\text{weight}(x|_{B_{s-1}}) \geq \frac{k}{2} - \sqrt{\frac{1}{2} dk \log n} \;\middle|\; \text{weight}(x|_{R_s}) = k \right] \geq 1 - \frac{2}{n^d}$$

For any integer r in the appropriate range:

$$\Pr_x \left[\text{weight}(x|_{B_{s-1}}) = r \;\mid\; \text{weight}(x|_{R_s}) = k \right] = \frac{\binom{M}{r}\binom{M}{k-r}}{\binom{2M}{k}} \tag{4}$$

where $M = dn^2 \log n$ is the size of B_{s-1}. Define Y to be the following random variable:

$$\Pr[Y = r] = \Pr_x \left[\text{weight}(x|_{B_{s-1}}) = r \;\mid\; \text{weight}(x|_{R_s}) = k \right]$$

Then by (4), Y is a hyper-geometric random variable. We need to bound

$$\Pr \left[Y \geq k/2 - \sqrt{dk \log n/2} \right]$$

from below. 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 $a > 0$

$$\Pr \left[\left| Y - \frac{k}{2} \right| < a \right] \geq \Pr \left[\left| X - \frac{k}{2} \right| < a \right] \geq 1 - 2e^{-2a^2/k}$$

where the last inequality follows from the standard Chernoff bound [1, Appendix A]. Plugging in $a = \sqrt{dk \log n/2}$ leads to

$$\Pr \left[Y \geq \frac{k}{2} - \sqrt{\frac{1}{2} dk \log n} \right] \geq \Pr \left[\left| Y - \frac{k}{2} \right| \leq \sqrt{\frac{1}{2} dk \log n} \right] \geq 1 - 2e^{-d \log n} = 1 - \frac{2}{n^d}$$

□

We now show that the achievable sets defined above satisfy the security condition of Theorem 4.2. Observe that the only set of coalitions of exactly $n-1$ users whose intersection is empty is the set of all n coalitions $U_1, \ldots, U_n$. Hence, all that needs to be shown is that $\bigcap_{s=1}^{n} A_{U_s} = \emptyset$.

Lemma 4.4 *For the code* $\Gamma_0(n, 2dn^2 \log n)$*, we have*

$$\bigcap_{s=1}^{n} A_{U_s} = \emptyset$$

The proof of the lemma relies on the following claim:

Claim 4.5 *Suppose* $x \in \bigcap_{s=1}^{n} A_{U_s}$*. Then for any* $1 \leq s \leq n-1$*:*

$$\mathit{weight}(x|_{B_s}) \leq 2ds^2 \log(n)$$

Proof The proof is by induction on s. For $s = 1$, the claim is immediate since $x \in A_{U_1}$ implying that $x|_{B_1}$ is all 0's, that is weight$(x|_{B_1}) = 0$.

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

$$k = \text{weight}(x_{B_s}) \quad k' = \text{weight}(x_{B_{s+1}}) \quad t = \text{weight}(x_{R_{s+1}})$$

Then the following three conditions are satisfied:

$$\begin{aligned} t &= k + k' \\ k &\leq 2ds^2 \log n \\ k &\geq \frac{t}{2} - \sqrt{\frac{1}{2} dt \log n} \end{aligned}$$

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 $x \in A_{U_{s+1}}$. We show that these three conditions imply $k' \leq 2d(s+1)^2 \log n$ which will prove the claim for $s+1$.

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

which leads to

$$k' \leq 2ds^2 \log n + \sqrt{2d(2ds^2 \log n + k') \log n}$$

Suppose $k' = 2dr^2 \log n$ for some constant r. Substituting for k' and dividing by $2d \log n$ 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, $k' \leq 2d(s+1)^2 \log n$. □

Proof of Lemma 4.4 Assume towards a contradiction that $x \in \bigcap_{s=1}^{n} A_{U_s}$. Since $x \in A_{U_n} = F(U_n)$, we know that $x|_{B_{n-1}}$ must be all 1's[7]. Since the size of the block B_{n-1} is $2dn^2 \log(n)$ we know that

$$\text{weight}(x|_{B_{n-1}}) = 2dn^2 \log(n)$$

On the other hand, for $s = n-1$ Claim 4.5 bounds the weight of $x|_{B_{n-1}}$ by

$$\text{weight}(x|_{B_{n-1}}) \leq 2d(n-1)^2 \log n$$

This contradiction proves that $\bigcap_{s=1}^{n} A_{U_s} = \emptyset$. □

Thus Theorem 4.1 is proved. We conclude this section by proving a simple lower bound on the length of n-secure codes.

Theorem 4.6 *Any (n, l)-code which is n-secure with error ϵ must have length $l \geq n-1$, for any $\epsilon > 0$.*

Proof Let Γ be an (n, l)-code which is n-secure with error $\epsilon > 0$. Let U_i be the coalition of $n-1$ users which doesn't contain user i. The fact that Γ is n-secure implies that to every U_i we can assign a set A_{U_i} which is ϵ-achievable for U_i and $\bigcap_{i=1}^{n} A_{U_i} = \emptyset$. We show that this condition implies that the code must have length $l \geq n-1$.

Suppose the length of the code Γ is less than $n-1$. There must exist a k, $1 \leq k \leq n-1$, such that there is no column in Γ of weight k. Consider the word $w \in \{0,1\}^l$ where the i'th bit of w is 0 if the i'th column in Γ has weight less than k, and is 1 otherwise. Since there is no weight k column, for every $i = 1, \ldots, n$, the word w is fixed throughout all the U_i blocks. Hence, w must be achievable for every U_i, that is $w \in \bigcap_{i=1}^{n} A_{U_i}$, contradicting the fact that the intersection is empty. □

4.2 Poly-log length c-secure codes

We use a similar method to that used in Section 2.1 to construct c-secure codes of length $c^{O(1)} \log^{O(1)} n$. The construction will be performed in two steps: First we show that an $\hat{F}$ totally secure code[8] can be used to extend a small c-secure code to a large one. Second, we construct an $\hat{F}$ totally c-secure code.

To simplify the notation in this section we identify users with the codewords assigned to them. That is, given a code Γ, we regard a user as a codeword of Γ. We regard a coalition of users as a subset of Γ.

Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ be a (p, l)-code which is c-secure with ϵ-error. Let $\hat{\Gamma}$ be a $(N, L)_p$-code which is $\hat{F}$ totally c-secure. We construct a (N, lL)-code Γ' as in Section 2.1. That is,

[7] Recall that the coalition U_n can not detect the bit positions in B_{n-1} since they see all 1's there. Hence, any string $y \in F(U_{n-1})$ must be all 1's in the block B_{n-1}.

[8] $\hat{F}$ totally secure codes were defined in Definition 3.3.

to every word $v \in \hat{\Gamma}$ we associate the word

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

The code Γ' is the set of all such words. We show that this resulting code is c-secure.

First we discuss how achievable sets behave when codes are concatenated. For two sets of words A_1, A_2, we denote by $A_1 \times A_2$ the set

$$A_1 \times A_2 = \{x \| y \ \mid \ x \in A_1 \ , \ y \in A_2\}$$

Lemma 4.7 *Let $\Gamma_1 = \{\alpha_1, \dots, \alpha_n\}$ and $\Gamma_2 = \{\beta_1, \dots, \beta_m\}$ be two codes. Consider a coalition $U \subseteq \Gamma_1 \times \Gamma_2$. Then*

$$U = \{\alpha_{i_1} \| \beta_{j_1}, \dots, \alpha_{i_c} \| \beta_{j_c}\}$$

Define $U_1 = \{\alpha_{i_1}, \dots, \alpha_{i_c}\}$ and $U_2 = \{\beta_{j_1}, \dots, \beta_{j_c}\}$. If A_1 is ϵ_1-achievable for U_1 and A_2 is ϵ_2-achievable for U_2 then $A = A_1 \times A_2$ is $(\epsilon_1 + \epsilon_2)$-achievable for U.

Proof Let $y \in F(U; \Gamma_1 \times \Gamma_2)$. We show that

$$\Pr_x[x \in A \ \mid \ x \in \mathrm{SYM}(y, U)] > 1 - (\epsilon_1 + \epsilon_2)$$

The core point here is that

$$\mathrm{SYM}(y, U) = \bigcup \mathrm{SYM}(y_1, U_1) \times \mathrm{SYM}(y_2, U_2)$$

where the union is over all y_1, y_2 such that $y_1 \| y_2 \in \mathrm{SYM}(y, W)$. Furthermore, this is a partition of $\mathrm{SYM}(y, U)$ since for y_1, y_2, y_1', y_2', the sets $\mathrm{SYM}(y_1, U_1) \times \mathrm{SYM}(y_2, U_2)$ and $\mathrm{SYM}(y_1', U_1) \times \mathrm{SYM}(y_2', U_2)$ are either identical or disjoint.

We know that for any $y_1 \in F(U_1; \Gamma_1)$ and $y_2 \in F(U_2; \Gamma_2)$ we have

$$\Pr_x[x \in A_1 \ \mid \ x \in \mathrm{SYM}(y_1, U_1)] > 1 - \epsilon_1$$

$$\Pr_x[x \in A_2 \ \mid \ x \in \mathrm{SYM}(y_2, U_2)] > 1 - \epsilon_2$$

Hence,

$$\begin{aligned} \Pr_x[x \in A \ \mid \ & x \in \mathrm{SYM}(y_1, U_1) \times \mathrm{SYM}(y_2, U_2)] = \\ & \Pr_{x_1, x_2}[x_1 \in A_1 \text{ and } x_2 \in A_2 \ \mid \ x_1 \in \mathrm{SYM}(y_1, U_1) \text{ and } x_2 \in \mathrm{SYM}(y_2, U_2)] \\ > \ & (1 - \epsilon_1)(1 - \epsilon_2) > 1 - (\epsilon_1 + \epsilon_2) \end{aligned}$$

Since this holds for all members of the partition, it holds for all of $\mathrm{SYM}(y, U)$. This proves that the set A is ϵ-achievable for the coalition U. □

We now prove that the code Γ' defined above is c-secure.

Theorem 4.8 *The code Γ' is c-secure with error ϵL.*

Proof Let $U = \{u_1, \ldots, u_r\}$ be a coalition of at most c users. For each $u_i \in U$ let $v^{(i)} \in \hat{\Gamma}$ be the codeword from which u_i's word in Γ' was derived. For every $j = 1, \ldots, L$ define

$$P_j(U) = \{v_j^{(1)}, \ldots, v_j^{(r)}\}$$

The elements of $P_j(U)$ are letters from an alphabet of size p. Hence, the set $P_j(U)$ can be thought of as a coalition of users of the (p, l)-code Γ. Since the code Γ is c-secure, every coalition P of at most c users has an ϵ-achievable set A_P associated with it. We define the achievable set A'_U of U as follows:

$$A'_U = A_{P_1(U)} \times A_{P_2(U)} \times \cdots \times A_{P_L(U)}$$

To complete the proof we need to show two things: the sets defined above are ϵL-achievable for U and they satisfy the security condition.

The fact that the sets A'_U are ϵL-achievable follows from Lemma 4.7 by applying it L times.

We now show that the sets A'_U defined above satisfy the security condition. Let $U_1, \ldots, U_k$ be a set of coalitions of at most c users each. Suppose $x \in \bigcap_{j=1}^{k} A'_{U_j}$. The word x can be broken down into L blocks $x_1, \ldots, x_L$ corresponding to the L blocks of Γ comprising Γ'. By definition of A'_{U_j}, we know that for all $j = 1, \ldots, L$ we have

$$x_j \in \bigcap_{i=1}^{L} A_{P_j(U_i)}$$

Hence, the set $\bigcap_{i=1}^{L} A_{P_j(U_i)}$ is not empty. Since the achievable sets of Γ satisfy the security condition, we know that $\bigcap_{i=1}^{L} P_j(U_i) \neq \emptyset$ for all j. Hence, for each $j = 1, \ldots, L$, it is possible to pick some

$$y_j \in \bigcap_{i=1}^{L} P_j(U_i)$$

Consider the word $y = (y_1, \ldots, y_L)$. Observe that for all coalitions $U_1, \ldots, U_k$, the word y satisfies

$$y \in \hat{F}(U_j; \hat{\Gamma})$$

Hence, $\bigcap_{j=1}^{k} \hat{F}(U_j; \hat{\Gamma})$ is not empty. Since the code $\hat{\Gamma}$ is $\hat{F}$ totally c-secure, we conclude that

$$\bigcap_{i=1}^{k} U_i \neq \emptyset$$

which is what we wanted to show. □

In order to use Theorem 4.8, we must construct $\hat{F}$ totally secure codes. This is achieved using error correcting codes with large minimal distance.

Lemma 4.9 *An $(L, N, D)_p$ error correcting code $\mathcal{C}$ is $\hat{F}$ totally c-secure provided*

$$D > L(1 - \frac{1}{c^2})$$

Proof Let $U_1, \ldots, U_r$ be a set of coalitions each with at most c users. Suppose $x \in \bigcap_{i=1}^{r} \hat{F}(U_i)$. We show that the intersection of the U_i's is not empty.

By definition, Since $x \in \hat{F}(U_1)$, we know that in every position, x must match a codeword of some user in U_1. Since there are at most c users in U_1, there must exist a user $u \in U_1$ whose codeword $w \in \mathcal{C}$ matches x in at least L/c positions. Let I be the set of positions where w matched x, that is $x|_I = w|_I$. We show that for any coalition U of at most c users, if $x \in \hat{F}(U)$, then $u \in U$. This implies $u \in \bigcap_{i=1}^{r} U_i$ and thus the intersection of the U_i's is not empty.

Suppose $x \in \hat{F}(U)$ but $u \notin U$. Since $x \in \hat{F}(U)$, there must exist a user $u' \in U$ whose codeword w' matches $x|_I$ in at least $|I|/c$ positions. Since $x|_I = w|_I$ and $|I| \geq L/c$, we see that w must match w' in at least L/c^2 positions. However, this implies that the Hamming distance between w and w' is at most $L(1-1/c^2)$ contradicting the hypothesis of the lemma. □

We summarize the construction in the following theorem:

Theorem 4.10 *Let Γ be a (p, l)-code which is c-secure with error ϵ and let $\mathcal{C}$ be a $(L, N, D)_p$-ECC where $D > L(1 - 1/c^2)$. Then there exists an explicit (N, lL)-code which is c-secure with error ϵL.*

This theorem is analogous to Theorem 2.4. The code $\mathcal{C}(p, d)$ constructed in Lemma 2.5 has a large minimal distance and can be used as the ECC in Theorem 4.10. Recall that the code $\mathcal{C}(p, d)$ has parameters $(p, p^{d+1}, p - d)_p$.

We now combine all our results to construct a code for n users which is c-secure with $1/n$-error. Set

$$p = c^2 \log n \quad d = \log n - 1$$

The code $\mathcal{C}(p, d)$ will contain more than n words and will satisfy the criterion of Theorem 4.10 since $p - d > p(1 - 1/c^2)$.

By Theorem 4.1 the code $\Gamma_0(p, p^3 \log p)$ will be a code for p users which is p-secure with error $\left(\frac{1}{2}\right)^{p+\log p} < 1/np$. The length of the code will be $O(p^4 \log p)$.

By Theorem 4.10 we can now use the codes $\mathcal{C}(p, d)$ and $\Gamma_0(p, p^3 \log p)$ to construct a code for n users which is c-secure with error $1/n$. The length of the code is

$$l = O(pp^4 \log p) = \tilde{O}(c^{10} \log^5 n)$$

where $\tilde{O}$ means we ignore $\log \log(n)$ factors. We have proved:

Corollary 4.11 *For any n, c, there are explicit codes for n users of length $\tilde{O}(c^{10} \log^5 n)$ which are c-secure with error $1/n$.*

A reasonable choice for c is $c = \log n$. In this case the length of the code becomes poly-logarithmic in the number of users. Note that this is an asymptotic result. Without ignoring constants, the length of these codes is shorter and practical.

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 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.

For example, a user might receive a disk D containing an encrypted executable. The disk D will be shipped to any user who decides to purchase the software. The user would then contact the distributor to get a small number of additional bits, K. The pair (D, K) will be used to create a usable executable which is fingerprinted with the user's codeword. The distribution scheme should be secure in the sense that the pair (D, K) should not enable the user to create copies of the executable marked with different fingerprints.

We call the disk D the *public data* and the key K the *private data*. In our concern about distribution, we are making the implicit assumption that it is less costly to distribute public data than private data.

Throughout this section, let Γ be an (n, l)-code which is used to fingerprint the objects. We denote the object to be distributed by P and let L be its length. We describe a distribution scheme in which the public data has length $2L$ and the private data has length $O(l)$.

Partition the object P into l pieces. We will embed exactly one mark in each piece. We assume that each piece is of sufficient size and complexity to satisfy the Marking Axiom. Hence, a single user can not detect the marks embedded in the pieces. 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. Clearly, if we wish to distribute the copy with codeword x to a user u, it is sufficient to send him his partition set $P(x)$. The distribution problem then becomes stating a method to send $p(1, x_1), \dots, p(l, x_l)$ to each user such that no user u is able to infer any partition $p(r, s) \notin P(x)$.

We use a secure private key cryptosystem to encode each piece $p(r, s)$. See [5] for the precise definitions. Such systems are known to exist under some standard hardness assumptions [4]. 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 nonnegligible probability. This property is crucial for our purposes.

The distribution scheme should now be clear. For each r, $1 \leq r \leq l$, and $s \in \{0,1\}$, pick a random key $k(r,s)$. Let

$$q(r,s) = f_{k(r,s)}\Big(p(r,s)\Big) \quad \text{for} \quad 1 \leq r \leq l,\ s \in \{0,1\}$$

Thus, each piece is encrypted with its own unique key. The public data, D, given to all users is

$$D = \Big\{q(r,s) \ \mid \ 1 \leq r \leq l,\ s \in \{0,1\}\Big\}$$

The size of D is twice the size of the original object, since every piece of the original object appears twice in D.

Let $x \in \Gamma$ be the word associated with user u. The private data given to user u will be the collection of keys that will enable him to retrieve $P(x)$ from the public data D. Define

$$h_u = \{k(1,x_1), k(2,x_2), \ldots, k(l,x_l)\}$$

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

The following lemma shows that this method is secure. For a user u, let P_u be the copy of the object which is fingerprinted with the codeword assigned to user u.

Lemma 5.1 *Let $C = \{u_1, \ldots, u_c\}$ be a coalition of users. Suppose there exists a polynomial time algorithm A such that given $D, h_{u_1}, \ldots, h_{u_c}$ finds an undetectable mark for C. Then there exists a polynomial time algorithm A' which given $P_{u_1}, \ldots, P_{u_c}$ finds an undetectable mark for C.*

Proof Let $x^{(1)}, \ldots, x^{(c)} \in \Gamma$ be the codewords assigned to the users in C. Let $1 \leq j \leq l$ be some undetectable bit position for C, that is $j \in [C;\Gamma]$. Without loss of generality, assume that all the users in C see a 1 in position j, i.e., $x_j^{(1)} = \ldots = x_j^{(c)} = 1$. The coalition does not have the key $k(j,0)$ implying that they can not recover $p(j,0)$ from $q(j,0) \in D$.

The fact that f_k is a secure private key cryptosystem, implies that there exists an algorithm A_0 which receives as input $D - \{q(j,0)\}, h_{u_1}, \ldots, h_{u_c}$ and still recovers an undetectable mark for C. We can iterate this argument and end up with an algorithm A' which receives as input

$$\bigcup_{j=1}^{c} P(x^{(j)}), h_{u_1}, \ldots, h_{u_c}$$

and still finds an undetectable mark for C. This is equivalent to running the algorithm with the input $P_{u_1}, \ldots, P_{u_c}$. □

The lemma implies that if users can detect marks using the data provided by the distribution scheme, they can also detect them using the fingerprinted objects they get. Hence, the distribution scheme does not leak any information regarding the location of the fingerprints. We have proved the following theorem.

Theorem 5.2 *Let P be an object of size L and Γ an (n,l)-code which is used to fingerprint P. It is possible to solve the distribution problem using public data of size $2L$ and private data of size $O(lL^{1/m})$ for some fixed constant $m > 1$.*

Note that it is crucial that the private key cryptosystem f_k not leak any bits. To see this, suppose that, for instance, $f_k(x)$ leaked the fact that the first and second bits of x have the same parity. Using our distribution scheme, a user sees $p(j,0)$ and $q(j,1) = f_k(p(j,1))$ or vice versa. Now, suppose that given $q(j,1)$ the user could determine that the first and second bits of $p(j,1)$ have the same parity. Then if the user discovers that in $p(j,0)$ the first and second bits have different parity, he can deduce that the mark is implemented in either the first or second bits.

There are several variants to this distribution scheme which will decrease the size of the private data at the expense of increasing the public data. Suppose that instead of embedding one mark per partition, we embed t marks in every partition. Then, using the same idea as above, the size of the public data required becomes $2^t L$ while the size of the private data reduces to Kl/t.

Another variant of the above scheme is to group the keys $k(r,s)$ together. Let d be some parameter which will be set later. For simplicity assume d divides l where l is the length of the code Γ. For every $i = 1, \ldots, l/d$ and $\bar{s} = s_1 s_2 \ldots s_d \in \{0,1\}^d$ define

$$\bar{k}(i,\bar{s}) = \Big\{k((i-1)d+1, s_1), k((i-1)d+2, s_2), \ldots, k((i-1)d+d, s_d)\Big\}$$

Thus, $\bar{k}(i,\bar{s})$ is a group of keys chosen according to the bits of $\bar{s}$. We will use a secure private key cryptosystem to encrypt each block $\bar{k}(i,\bar{s})$. Let $g_k : \{0,1\}^{Kd} \to \{0,1\}^{Kd}$ be a secure private key cryptosystem. The key length of this cryptosystem is $\bar{K} = (Kd)^{1/m'}$ for some fixed constant m'. For every $i = 1, \ldots, l/d$, and $\bar{s} \in \{0,1\}^d$, pick a random key $e(i,\bar{s}) \in \{0,1\}^{\bar{K}}$. Define

$$\bar{q}(i,\bar{s}) = g_{e(i,\bar{s})}(\bar{k}(i,\bar{s}))$$

Define the new public data $\bar{D}$ to be

$$\bar{D} = D \bigcup \Big\{\bar{q}(i,\bar{s}) \;\mid\; 1 \le i \le \frac{l}{d},\ \bar{s} \in \{0,1\}^d\Big\}$$

where D is the public data in the original scheme. The size of $\bar{D}$ is $2L + 2^d lK$. Hence, d can be chosen to be $O(\log L)$ to make the size of the additional part negligible in comparison to the size of the original D.

Let $x \in \Gamma$ be the word assigned to user u. The private data for user u now becomes

$$\bar{h}_u = \{e(j, x_{(j-1)d} x_{(j-1)d+1} \ldots x_{jd-1}) \;\mid\; 1 \le j \le l/d\}$$

Given $(\bar{D}, \bar{h}_u)$, user u can build a usable copy of the object. This will be done by first recovering the $k(r,s)$ from $\bar{q}(i,\bar{s})$ using $\bar{h}_u$. Then, recover $p(r,s)$ from $q(r,s)$ using $k(r,s)$ as before.

We have reduced the size of the private data to $K'(l/d)$. Choosing $d = O(\log L)$ reduces the size of the private data by a factor of $\log L$ while increasing the size of the public data by some constant factor. We summarize this in the following theorem.

Theorem 5.3 *Let P be an object of size L and Γ an (n, l)-code which is used to fingerprint P. It is possible to solve the distribution problem using public data of size $O(L)$ and private data of size $O(L^{1/m}\frac{l}{\log L})$ for some fixed constant $m > 1$.*

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 gets rid of the $L^{1/m}$ factor in the size of the private data, which 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 $\log^4 n$. For the general redistribution problem considered in Section 3, the length of our codes was poly-logarithmic in 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 Axiom".

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. Thus, 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 encoding marks can be used to fingerprint movies, we say that the Marking Axiom is satisfied in the domain of movies.

Showing that the Marking Axiom is satisfied for software is much harder. Coming up with a formal model for marking software and describing a practical method for implementing marks is an important problem. 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?

The c-secure codes constructed in the previous section are quite long. A natural problem is to try to construct shorter codes. We have no good lower bounds on the length of c-secure codes. It is interesting to note that random codes do very badly as c-secure codes. The reason is that the length of the code must be exponential in c. To see this, observe that if the

length of the code was less than 2^c, then a coalition of c users is likely to detect all the bits. The coalition can thus replace the fingerprint with a truly random fingerprint, providing no information.

For the sake of completeness, we briefly discuss an alternate method of fingerprinting which is only applicable when fingerprinting software. The core problem of fingerprinting is to prevent users from changing their fingerprints in a way that masks their identities. For software, this can be achieved using one way functions and the notion of *locks*. A lock is a routine which is executed by the program. The lock routine will check that the fingerprint is valid, and if it isn't, the program will fail. Using one-way functions, it is possible to ensure that no user can create a valid fingerprint which is different from his own. The same will hold for any coalition of users. Hence, the users can not change the fingerprint without making the program useless. We refer to this scheme as the *active scheme*. Clearly, the active scheme can not be applied to domains such as books, movies, etc. since there is no notion of a lock.

For the active scheme to be secure, the users must not be able to detect the lock routine. If they do detect it, they can remove it and then freely change the fingerprint. There is some empirical evidence to support the claim that lock routines can not be hidden in executables. Hence, we claim that the active scheme is not a secure method of implementing fingerprints.

The active scheme is fundamentally different from the fingerprinting method we have been considering. The difference is that the program is actively checking that the fingerprint is valid. In contrast, our standard notion is that a fingerprint is embedded in the executable and never explicitly referenced by the program. We say that our fingerprints are *passive*. The assumption that hiding passive marks in software is possible is weaker than the assumption of hiding locks. Hence, our approach is more secure when doing software protection.

Acknowledgments

The authors wish to thank Richard Lipton for some discussions on this work.

References

[1] N. Alon and J. Spencer, *The probabilistic method*, Wiley, 1992.

[2] 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.

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

[4] M. Luby and C. Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*, SIAM Journal of computing, Vol. 17, 1988, pp. 373-386

[5] S. Micali, C. Rackoff and B. Sloan, *The notion of security for probabilistic cryptosystems*, SIAM Journal of computing, Vol. 17, 1988, pp. 412-426.

[6] O. Goldreich and R. Ostrovsky, *Software protection and simulation on oblivious RAMs*, to appear.

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

[8] Software Publishers Association, Press release, 1994.

[9] 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.

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

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