Traitor Tracing with Constant Size Ciphertext

Dan Boneh*   Moni Naor

February 22, 2008

Abstract

A traitor tracing system enables a publisher to trace a pirate decryption box to one of the secret keys used to create the box. We present the first traitor tracing system where ciphertext size is "constant," namely independent of the number of users in the system and the collusion bound. A ciphertext in our system consists of only two elements where the length of each element depends only on the security parameter. The down side is that private-key size is quadratic in the collusion bound. Our construction is based on recent constructions for fingerprinting codes.

1 Introduction

Traitor tracing systems, introduced by Chor, Fiat, and Naor [7], help content distributors identify pirates who violate copyright restrictions. To be concrete, consider a satellite radio system (such as XM Satellite Radio) where broadcasts should only be played on certified radio receivers. We let n denote the total number of radio receivers and assume that each receiver contains a unique secret key: radio receiver number i contains secret key sk_i. Broadcasts are encrypted using a broadcast key bk and any certified receiver can decrypt using its secret key. Certified players, of course, can enforce digital rights restrictions such as "do not copy" or "play once".

Clearly a pirate could hack a number of certified players and extract their secret keys. The pirate could then build a pirate decoder PD that will extract the cleartext content and ignore any relevant digital rights restrictions. Even worse, the pirate can make its pirate decoder widely available so that anyone can extract the cleartext content for themselves. DeCSS, for example, is a widely distributed program for decrypting encrypted DVD content.

This is where traitor tracing is helpful — when the pirate decoder PD is found, the distributor can run a *tracing* algorithm that interacts with the pirate decoder and outputs the index i of at least one of the keys sk_i that the pirate used to create the pirate decoder. The distributor can then choose to take action against the owner of this sk_i.

A precise definition of traitor tracing systems is given in [3] and is reproduced here in Appendix A. For now we give some intuition that will help explain our results. A traitor tracing system consists of four algorithms *Setup*, *Encrypt*, *Decrypt*, and *Trace*. The *Setup* algorithm generates the broadcaster's key bk, a tracing key tk, and n recipient keys $sk_1, \dots, sk_n$. The *Encrypt* algorithm encrypts the content using bk and the *Decrypt* algorithm decrypts using one of the sk_i. The tracing algorithm *Trace* is the most interesting — it takes tk as input and interacts with a

*Supported by NSF and the Packard Foundation.

pirate decoder, treating it as a black-box oracle. It outputs the index $i \in \{1, \ldots, n\}$ of at least one key sk_i that was used to create the pirate decoder.

We will describe our system as a public-key system, namely we assume that bk is public. As in many traitor tracing constructions, the tracing key tk in our system must be kept secret. Our tracing algorithm is black-box, namely it need not look at the internals of the pirate decoder PD. The tracing algorithm only interacts with PD as if it were a decryption oracle.

a traitor tracing system is said to be t-collusion resistant if tracing will work as long as the pirate has fewer than t user keys at his disposal. If $t = n$ the system is said to be *fully collusion resistant*. While ciphertext-size in our system is independent of n or t, private-key size is quadratic in t. More precisely, our system provides the following parameters as a function of the total number of users n, collusion bound t, and security parameter λ:

$$\text{CT-len} = O(\lambda) \quad ; \quad \text{SK-length} = O(t^2\lambda^2 \log n) \quad ; \quad \text{Tracing-time} = O(t^2\lambda \log n)$$

Setting $t \leftarrow n$ gives the parameters for full collusion resistance. Note that ciphertext length is independent of n or t.

Related work. Traitor tracing systems have been studied extensively. We refer to [3] for various properties of traitor tracing systems. Traitor tracing constructions generally fall into two categories: combinatorial, as in [7, 21, 28, 29, 10, 11, 8, 25, 1, 27, 26, 20], and algebraic, as in [18, 2, 22, 17, 9, 19, 31, 6, 3, 5]. The broadcaster's key bk in combinatorial systems can be either secret or public. Algebraic traitor tracing use public-key techniques and are often more efficient than the public-key instantiations of combinatorial schemes. In these systems the ciphertext length (for short messages) depends linearly on the collusion bound t. One exception is [3] which is fully collusion resistant with ciphertext size $O(\sqrt{n})$.

Some systems, including ours, only provide tracing capabilities. Other systems [22, 20, 14, 12, 9, 5] combine tracing with broadcast encryption to obtain trace-and-revoke features — after tracing, the distributor can revoke the pirate's keys without affecting any other legitimate decoder.

Kiayias and Yung [17] describe a black-box tracing system that achieves constant rate for long messages, where rate is measured as the ratio of ciphertext length to plaintext length. For full collusion resistance, however, the ciphertext size is linear in the number of users n. For comparison, our new system generates ciphertexts of constant size and achieves constant rate (rate = 1) for long messages by using hybrid encryption (i.e. encrypting a short message-key using the traitor tracing system and encrypting the long data by using a symmetric cipher with the message-key).

In most traitor tracing systems, including ours, the tracing key tk must be kept secret. Some systems, however, support public key tracing [23, 24, 32, 16, 6].

Stateful vs. Stateless decoders: a stateless decoder is one that does not keep state between decryptions. For instance, software decoders, such as DeCSS, cannot keep any state. However, pirate decoders embedded in tamper resistant hardware, such as a pirate cable box, can keep state between successive decryptions. When the decoder detects that it is being traced it could shutdown and refuse to decrypt further inputs. A software decoder cannot do that. Kiayias and Yung [15] show how to convert any tracing system for stateless decoders into a tracing system for stateful decoders by embedding robust watermarks in the content. Consequently, most tracing systems in the literature, as do we, focus on the stateless settings and ignore the stateful case.

2 Collusion resistant fingerprinting codes

Before describing our construction we first review the definition of collusion resistant fingerprinting codes from [4]. Collusion resistant codes are primarily used for fingerprinting digital content. Here we will use them to construct a traitor tracing system with short ciphertexts. We are only interested in *binary codes*, namely codes defined over $\{0,1\}$ (as opposed to a larger alphabet).

- For a word $\bar{w} \in \{0,1\}^\ell$ we write $\bar{w} = w_1 \dots w_\ell$ where $w_i \in \{0,1\}$ is the ith letter of $\bar{w}$ for $i = 1, \dots, \ell$.
- Let $W = \{\bar{w}^{(1)}, \dots, \bar{w}^{(t)}\}$ be a set of words in $\{0,1\}^\ell$. We say that a word $\bar{w} \in \{0,1\}^\ell$ is **feasible** for W if for all $i = 1, \dots, \ell$ there is a $j \in \{1, \dots, t\}$ such that $\bar{w}_i = \bar{w}_i^{(j)}$. For example, if W consists of the two words:
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$
then all words of the form $\left[0\binom{0}{1}\binom{0}{1}1\binom{0}{1}\right]$ are feasible for W.
- For a set of words $W \subseteq \{0,1\}^\ell$ we say that the **feasible set** of W, denoted $F(W)$, is the set of all words that are feasible for W.

A **fingerprinting code** is a pair of algorithms (G, T) defined as follows:

- Algorithm G, called a **code generator** is a probabilistic algorithm that takes a pair (n, ϵ) as input, where n is the number of words to output and $\epsilon \in (0,1)$ is a security parameter. The algorithm outputs a pair (Γ, tk). Here Γ (called a **code**) contains n words in $\{0,1\}^\ell$ for some $\ell > 0$ (called the **code length**). The output tk is called the **tracing key**.
- Algorithm T, called a **tracing algorithm**, is a deterministic algorithm that takes as input a pair $(\bar{w}^*, tk)$ where $\bar{w}^* \in \{0,1\}^\ell$. The algorithm outputs a subset S of $\{1, \dots, n\}$. Informally, elements in S are "accused" of creating the word $\bar{w}^*$.

We require that G and T run in polynomial time in $n \log(1/\epsilon)$.

Security of a fingerprinting code (G, T) is defined using a game between a challenger and an adversary. Let n be an integer and $\epsilon \in (0,1)$. Let C be a subset of $\{1, \dots, n\}$. Both the challenger and adversary are given (n, ϵ, C) as input. Then the game proceeds as follows:

1. The challenger runs $G(n, \epsilon)$ to obtain (Γ, tk) where $\Gamma = \{\bar{w}^{(1)}, \dots, \bar{w}^{(n)}\}$. It sends the set $W := \{\bar{w}^{(i)}\}_{i \in C}$ to the adversary.
2. The adversary outputs a word $\bar{w}^* \in F(W)$.

We say that the adversary $\mathcal{A}$ wins the game if $T(\bar{w}^*, tk)$ is empty or not a subset of C.
Let CR Adv$[(G(n, \epsilon), T, C), \mathcal{A}]$ be the probability that $\mathcal{A}$ wins the game.

Definition 1. *We say that a a fingerprinting code (G, T) is **fully collusion resistant** if for all adversaries $\mathcal{A}$, all $n > 0$, all $\epsilon \in (0,1)$, and all subsets $C \subseteq \{1, \dots, n\}$, we have that*

$$\text{CR Adv}[(G(n, \epsilon), T, C), \mathcal{A}] < \epsilon$$

*We say that (G, T) is t-**collusion resistant** if for all adversaries $\mathcal{A}$, all $n > t$, all $\epsilon \in (0,1)$, and all subsets $C \subseteq \{1, \dots, n\}$ of size at most t, we have*

$$\text{CR Adv}[(G(n, \epsilon), T, C), \mathcal{A}] < \epsilon$$

2.1 Known results on collusion resistant codes

Boneh and Shaw [4] constructed a fully collusion resistant fingerprinting code as well as t-collusion resistant codes. G. Tardos [30] improved these results by constructing shorter codes. The resulting code lengths are summarized in the following table.

	Boneh-Shaw [4]	G. Tardos [30]
Full collusion resistance	$\ell = O\big(n^3 \log(n/\epsilon)\big)$	$\ell = O\big(n^2 \log(n/\epsilon)\big)$
t-collusion resistance	$\ell = O\big(t^4 \log(n/\epsilon) \log(1/\epsilon)\big)$	$\ell = O\big(t^2 \log(n/\epsilon)\big)$

ℓ is the length of the code obtained by running $G(n, \epsilon)$

Throughout the paper, except for Section 4, we will primarily rely on the construction of G. Tardos [30].

We note that Chor et al. [7] constructed collusion resistant codes; however, their codes are defined over a much larger alphabet, namely Γ is a subset of $\{1, \ldots, t\}^\ell$ rather than $\{0, 1\}^\ell$. For the application we have in mind it is crucial that we use a fingerprinting code defined over a *binary* alphabet. Other constructions over large alphabets include [28, 29, 11, 27, 26]

3 A traitor-tracing system with short ciphertexts

Let $\mathcal{E} := (G_{enc}, E_{enc}, D_{enc})$ be a public-key encryption system. Let (G_{tt}, T_{tt}) be a fingerprinting code. Our traitor tracing system TT works as follows:

Setup(n, λ)**:** Let $\epsilon := 1/2^\lambda$. The algorithm works as follows:

1. Generate a fingerprinting code by running $(\Gamma, tk) \xleftarrow{\text{R}} G_{tt}(n, \epsilon)$.
   Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\} \subseteq \{0, 1\}^\ell$.
2. Generate 2ℓ public/secret key pairs by running G_{enc} 2ℓ times:
   $$\text{for } i = 1, \ldots, \ell \text{ and } j = 0, 1 \text{ do:} \qquad (pk[i, j],\ sk[i, j]) \xleftarrow{\text{R}} G_{enc}(\lambda)$$
3. For $i = 1, \ldots, n$ define
   $$sk_i \leftarrow \big(\ \bar{w}^{(i)},\ \ sk[1, w_1^{(i)}],\ \ldots,\ sk[\ell, w_\ell^{(i)}]\ \big)\ .$$
   An example secret key is shown in Figure 1.
4. Define $\quad bk \leftarrow \big(pk[1, 0],\ pk[1, 1], \ldots, pk[n, 0],\ pk[n, 1]\big)$
5. Output bk, tk, and $(sk_1, \ldots, sk_n)$

Encrypt(bk, m)**:** Choose random $j \xleftarrow{\text{R}} \{1, \ldots, \ell\}$ and compute

$$c_0 \xleftarrow{\text{R}} E_{enc}(pk[j, 0], m), \qquad c_1 \xleftarrow{\text{R}} E_{enc}(pk[j, 1], m)$$

output $c \leftarrow (j, c_0, c_1)$. Note that the ciphertext is short.

$\bar{w}^{(i)}$	sk_i	
0	$sk[1,0]$	$sk[1,1]$
1	$sk[2,0]$	$sk[2,1]$
0	$sk[3,0]$	$sk[3,1]$
$\vdots$	$\vdots$	
0	$sk[\ell,0]$	$sk[\ell,1]$

Figure 1: An example secret key

Decrypt$\big(i,\ sk_i,\ (j, c_0, c_1)\big)$**:** if $w_j^{(i)} = 0$ output $D_{enc}(sk[j,0],\ c_0)$ else output $D_{enc}(sk[j,1],\ c_1)$.

The tracing algorithm: intuition. Suppose the adversary obtains a set of t secret keys and uses them to build a pirate decoder PD. We assume PD is a *useful* decoder, namely it correctly decrypts well-formed ciphertexts. The t keys at the adversary's disposal corresponds to t words in the fingerprinting code $\Gamma \subseteq \{0,1\}^\ell$. Let $C \subseteq \{0,1\}^\ell$ be the set containing these t words. Now, consider a particular $j \in \{1, \ldots, \ell\}$ and consider the *invalid* ciphertext

$$c^* := \big(j, \quad E_{enc}(pk[j,0],\ \mathrm{m}), \quad E_{enc}\big(pk[j,1],\ 0\big)\big)$$

Here m is some message not equal to 0. This ciphertext is invalid since we encrypted a different message under $pk[j,0]$ and $pk[j,1]$. Let us consider what happens when we run PD on c^*. We are interested in two cases.

- Case 1: Suppose all t codewords in C contain a 1 in position j. Then the adversary does not have $sk[j,0]$ and therefore $PD(c^*)$ will return a quantity different than m with high probability.

- Case 2: Suppose all t codewords in C contain a 0 in position j. Now the adversary does not have $sk[j,1]$ and therefore PD cannot distinguish c^* from a well-formed ciphertext. Consequently, $PD(c^*)$ will return m (otherwise PD is not a useful pirate decoder).

To make use of these two observations, let us define ℓ experiments, denoted by TR_j for $j = 1, \ldots, \ell$. Experiment TR_j is defined as follows: (we let $\mathcal{M}$ denote the finite message space of the public-key system $\mathcal{E}$)

$$\boxed{\begin{array}{ll} m \xleftarrow{\mathrm{R}} \mathcal{M} & \\ c_0 \xleftarrow{\mathrm{R}} E_{enc}\big(pk[j,0],\ \mathrm{m}\big), & c_1 \xleftarrow{\mathrm{R}} E_{enc}\big(pk[j,1],\ 0\big) \\ c^* \leftarrow (j, c_0, c_1) & \\ \hat{m} \leftarrow PD(c^*) & \end{array}}$$

Define $w_j \in \{0,1\}$ as follows:

$$w_j := \begin{cases} 0 & \text{if } m = \hat{m}, \text{ and} \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

The argument in Case 1 suggests that if all words in C have a 1 is position j then $w_j = 1$. The argument in Case 2 suggests that if all words in C have a 0 is position j then $w_j = 0$. It follows that the word

$$\bar{w}^* := w_1 \dots w_\ell \in \{0,1\}^\ell \tag{2}$$

is in the feasible set $F(C)$. But then running the tracing algorithm T_{tt} of the collusion resistant code on input $\bar{w}^*$ will output the identity of at least one of the words in C, which is also the identity of one of the keys in the pirate's possession.

The tracing algorithm. To make the intuition above rigorous, we spell out the tracing algorithm. The tracing algorithm $\mathit{Trace}^{PD}(tk)$ works as follows:

1. For each j in $\{1, \dots, \ell\}$ run experiment TR_j once.
2. Define the word $\bar{w}^*$ as in equations (1) and (2).
3. Output $T_{tt}(\bar{w}^*,\ tk)$.

Overall, the tracing algorithm makes a total of $O(\ell)$ calls to the pirate decoder PD. Using Tardos's t-collusion resistant code we have $\ell = O(t^2 \log(n/\epsilon)) = O(t^2 \lambda \log n)$ and therefore the total number of queries to PD is

$$\#\ PD \text{ queries} = O\big(t^2 \lambda \log(n)\big)$$

We note that this tracing algorithm is *minimal access* as defined at the end of appendix A. That is, the tracing algorithm does not need access to the decrypted message from PD. It only needs to know whether the ciphertext was decrypted correctly. This is a useful property when tracing pirate music players in practice — one only gets to observe whether the player plays the music or not.

3.1 Security

The following theorem shows that the traitor tracing system TT is t-collusion resistant, namely it satisfies the security definition in Appendix A. For the public-key system $\mathcal{E}$ and a semantic security adversary $\mathcal{B}$ we use SS Adv$[\mathcal{B}, \mathcal{E}]$ to denote $\mathcal{B}$'s advantage in winning the semantic security game against $\mathcal{E}$.

Theorem 1. *Suppose $\mathcal{E} = (G_{enc}, E_{enc}, D_{enc})$ is semantically secure and (G_{tt}, T_{tt}) is a t-collusion resistant fingerprinting code. Then TT is a t-collusion resistant traitor-tracing system.*

In particular, using the notation of Appendix A, for all $t > 0$, $n > t$, and all polynomial time adversaries $\mathcal{A}$, there exist semantic security adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ attacking $\mathcal{E}$ such that

$$\text{MH Adv}[\mathcal{A}, TT(n)](\lambda) \le (2\ell) \cdot \text{SS Adv}[\mathcal{B}_1, \mathcal{E}](\lambda)$$

$$\text{TR Adv}[\mathcal{A}, TT(n,t)](\lambda) \le \ell \cdot \text{SS Adv}[\mathcal{B}_2, \mathcal{E}](\lambda) + \epsilon + \frac{\ell}{|\mathcal{M}|}$$

where $\ell = O(t^2 \lambda \log n)$ and $\epsilon = 1/(2^\lambda)$.

The semantic security property (the bound on MH Adv$[\mathcal{A}, TT(n)]$ defined in Appendix A, Game 1) is immediate. We bound the adversary's advantage in winning the tracing game, namely TR Adv$[\mathcal{A}, TT(n,t)]$ defined in Appendix A, Game 2. This will follow from Lemma 2 below. For an adversary $\mathcal{A}$ in Game 2 we let $\bar{w}^*(\mathcal{A})$ denote the random variable representing the word $\bar{w}^*$ constructed in step 2 in the tracing algorithm while tracing a pirate decoder PD created by $\mathcal{A}$.

Lemma 2. *With the notation as in Theorem 1, let $C \subseteq \Gamma \subseteq \{0,1\}^\ell$ be the set of words corresponding to the set of private keys in the adversary's possession. Then for any adversary $\mathcal{A}$ in the tracing game (game 2) there exists a semantic security adversary $\mathcal{B}$ attacking $\mathcal{E} = (G_{enc}, E_{enc}, D_{enc})$ such that*

$$\Pr[\bar{w}^*(\mathcal{A}) \notin F(C)] \leq \ell \cdot \text{SS Adv}[\mathcal{B}, \mathcal{E}] + (\ell/|\mathcal{M}|)$$

Proof. Consider a modified tracing algorithm that produces a word $\bar{q}^*(\mathcal{A})$ as follows. For all $j = 1, \ldots, \ell$ run the following experiment:

$m \xleftarrow{\text{R}} \mathcal{M}$
if all words in C have a 1 in position j do:
$\quad c_0 \xleftarrow{\text{R}} E_{enc}\big(pk[j,0],\ \boxed{0}\big), \qquad c_1 \xleftarrow{\text{R}} E_{enc}\big(pk[j,1],\ \boxed{0}\big)$
else do:
$\quad c_0 \xleftarrow{\text{R}} E_{enc}\big(pk[j,0],\ \boxed{\text{m}}\big), \qquad c_1 \xleftarrow{\text{R}} E_{enc}\big(pk[j,1],\ \boxed{\text{m}}\big)$

$\hat{c}^* \leftarrow (j, c_0, c_1)$

$\hat{m} \leftarrow PD(\hat{c}^*)$

Define $q_j \in \{0,1\}$ as: $\quad q_j := \begin{cases} 0 & \text{if } m = \hat{m}, \text{ and} \\ 1 & \text{otherwise.} \end{cases}$
and $\bar{q}^*(\mathcal{A}) := q_1 \ldots q_\ell$.

We say that position j is **critical** for $\mathcal{A}$ if all words in C contain the same symbol at position j. We claim that $\Pr[w_j \neq q_j]$ must be negligible at all *critical* positions. In particular, for all critical positions $j \in \{1, \ldots, \ell\}$ there is a semantic security adversary $\mathcal{B}$ for $\mathcal{E}$ such that

$$\Pr[w_j \neq q_j] \leq \text{SS Adv}[\mathcal{B}, \mathcal{E}] \tag{3}$$

To see why, notice that when all bits at position j in C are 1 then $\mathcal{A}$ does not have $sk[j,0]$. However, if $\Pr[w_j \neq q_j]$ is non-negligible then $\mathcal{A}$ is able to distinguish $E_{enc}\big(pk[j,0],\ 0\big)$ from $E_{enc}\big(pk[j,0],\ m\big)$, which breaks semantic security of $\mathcal{E}$. A similar argument applies when all bits at position j are 0.

Let *bad* be the event that there exists some critical coordinate j for which $w_j \neq q_j$. It follows from (3) and the union bound that

$$\Pr[bad] \leq \ell \cdot \text{SS Adv}[\mathcal{B}, \mathcal{E}]$$

When event *bad* does not happen (i.e. $\bar{w}^*(\mathcal{A})$ and $\bar{q}^*(\mathcal{A})$ match at all critical positions) then $\bar{w}^*(\mathcal{A}) \in F(C)$ if and only if $\bar{q}^*(\mathcal{A}) \in F(C)$. Hence, we obtain that

$$\big|\Pr[\bar{w}^*(\mathcal{A}) \notin F(C)] - \Pr[\bar{q}^*(\mathcal{A}) \notin F(C)]\big| \leq \Pr[bad] \leq \ell \cdot \text{SS Adv}[\mathcal{B}, \mathcal{E}] \tag{4}$$

To complete the proof we argue that $\Pr[\bar{q}^*(\mathcal{A}) \notin F(C)] \leq \ell/|\mathcal{M}|$. There are two cases

- Consider a bit position j where all words in C have a 1 at position j. We argue that $q_j = 1$ with high probability. For this j, the ciphertext $\hat{c}^*$ does not depend on m and therefore running $PD(\hat{c}^*)$ will output m with probability at most $1/|\mathcal{M}|$. We conclude that for this j the probability that $q_j \neq 1$ is at most $1/|\mathcal{M}|$.

- Consider a bit position j where all words in C have a 0 in position j. We argue that $q_j = 0$. For this j, the ciphertext $\hat{c}^*$ is a valid encryption of m and, since PD is a useful decoder, $PD(\hat{c}^*)$ will output m with probability 1. Hence, q_j will always equal 0.

Summing over all bit positions we see that the probability that $\bar{q}^*(\mathcal{A})$ is inconsistent with C in any critical position is at most $\ell/|\mathcal{M}|$. It follows that

$$\Pr[\bar{q}^*(\mathcal{A}) \notin F(C)] \leq \ell/|\mathcal{M}| \tag{5}$$

Putting together equations (4) and (5) proves the lemma. $\square$

To complete the proof of Theorem 1 observe that when $\bar{w}^*(\mathcal{A}) \in F(C)$ then $T_{tt}(\bar{w}^*(\mathcal{A}),\ tk)$ outputs a member of C with probability at least ϵ. Hence,

$$\text{TR Adv}[\mathcal{A}, TT] \leq \ell \cdot \text{SS Adv}[\mathcal{B}, \mathcal{E}] + \epsilon + (\ell/|\mathcal{M}|)$$

as required. $\square$

4 Tracing imperfect pirate decoders

Our definition of secure traitor tracing in Appendix A requires that the pirate produce a perfect pirate decoder PD, namely a decoder that correctly decrypts all well-formed ciphertexts. In reality, the pirate may be content with a decoder PD than works only a fraction of the time, say decrypts only 10% of well-formed ciphertexts. When our tracing algorithm interacts with such a decoder it may produce a word $\bar{w}^*$ that is not in the adversary's feasible set $F(C)$ and consequently the fingerprinting code may fail to trace.

For a given broadcast key bk, let δ be the probability that PD fails to decrypt well-formed ciphertexts:

$$\delta := \Pr[m \stackrel{\text{R}}{\leftarrow} \mathcal{M},\ c \stackrel{\text{R}}{\leftarrow} \mathit{Encrypt}(bk, m)\ :\ PD(c) \neq m]$$

We call δ the **error-rate of** PD. Until now we focused on low-error pirates, namely when $\delta = 0$.

To address imperfect decoders we define a more sophisticated tracing algorithm. For $j = 1, \ldots, \ell$ define the following experiment denoted $\mathsf{RobustTR}_j$:

```
repeat the following steps λ ln ℓ times:
    m ←R M
    c0 ←R E_enc(pk[j,0], m),        c1 ←R E_enc(pk[j,1], 0)
    c* ← (j, c0, c1)
    m̂ ← PD(c*)
let p_j be the fraction of times that m = m̂

repeat the following steps λ ln ℓ times:
    m ←R M
    c0 ←R E_enc(pk[j,0], m),        c1 ←R E_enc(pk[j,1], m)
    c ← (j, c0, c1)
    m̂ ← PD(c)
let q_j be the fraction of times that m = m̂
```

Define $w_j \in \{0,1\}$ as:

$$w_j := \begin{cases} 0 & \text{if } p_j > 0 & \text{(then } \mathcal{A} \text{ must know } sk[j,0]) \\ 1 & \text{if } p_j = 0 \text{ and } q_j > 0 & \text{(then } \mathcal{A} \text{ must know } sk[j,1]) \\ \text{'?'} & \text{otherwise} & \text{(if } p_j = q_j = 0) \end{cases}$$

and set $\bar{w}^* := w_1 \dots w_\ell$. The symbol '?' at position j indicates that PD refuses to decrypt all ciphertexts created for coordinate j. We know nothing about $\mathcal{A}$'s knowledge of keys at this position.

If the error-rate δ satisfies $\delta < (1/\ell) - (1/\lambda)$ then for all $j = 1, \dots, \ell$ we expect that $q_j > 0$ and therefore $\bar{w}^*$ will contain no '?' symbols. In this case $\bar{w}^*$ is in the adversary's feasible set $F(C)$ and consequently running $T_{tt}(tk, \bar{w}^*)$ will correctly identify one of the pirates in C. Consequently we obtain a tracing algorithm that traces the pirate decoder PD as long as

$$\delta < (1/\ell) - (1/\lambda).$$

4.1 Tracing imperfect decoders

To trace a decoder with high error-rate $\delta \in [0,1)$ one needs a better fingerprinting code that can trace words containing the '?' symbol. A simple counting argument shows that, with high probability, the maximum fraction of '?' symbols in $\bar{w}^*$ is close to δ; otherwise PD cannot correctly decrypt a δ fraction of well-formed ciphertexts. Therefore, we need a fingerprinting code that can trace words in $\{0,1,?\}^\ell$ that contain up to $\delta \cdot \ell$ question marks.

We extend the definition of collusion resistant fingerprinting from Section 2 to handle '?' symbols. First, for a set of words $W \subseteq \{0,1\}^\ell$ we say that a word $\bar{w} \in \{0,1,?\}^\ell$ is feasible for W if it is feasible for W when one considers only the non-'?' coordinates. We say that the **extended feasible set** for W, denoted $F_?(W)$, is the set of all feasible words for W in $\{0,1,?\}^\ell$.

Informally, we say that a fingerprinting code is **δ-robust** if the tracing algorithm can trace a word $\bar{w}^* \in \{0,1,?\}^\ell$ that is feasible for a subset C and contains at most $\delta \cdot \ell$ symbols '?', back to a member of C. More precisely, we modify step 2 in the game used in Definition 1 as follows:

2. The adversary outputs a word $\bar{w}^* \in F_?(W)$ that contains at most $\delta \cdot \ell$ symbols '?'.

We let CR $\text{Adv}[(G(n,\epsilon,\delta), T, C), \mathcal{A}]$ be the probability that $\mathcal{A}$ wins the game and this quantity is used in Definition 1. We say that the fingerprinting code is **δ-robust** if it satisfies Definition 1 for this value of δ.

Robust fingerprinting codes. It remains to construct robust fingerprinting codes. There are two relevant results in this direction, but unfortunately neither one is adequate:

- Boneh and Shaw [4] allow '?' symbols in the word $\bar{w}^*$, but only at non-critical coordinates (i.e. coordinates where C is not all 0 or all 1). In our case, however, PD is free to cause a '?' symbol to appear in any coordinate. As a result, this extension is not helpful here.

- Guth and Pfitzmann [13] consider fingerprinting codes where a δ-fraction of the coordinates in $\bar{w}^*$ are '?'. The '?' locations, however, must be chosen independently of the attacker's view. In our case, $\mathcal{A}$ can instruct PD to adversarialy choose the location of '?' symbols. For example, the adversary may cause a '?' to appear at all critical coordinates. Consequently, this extension is not helpful either.

In the context of fingerprinting digital content, there was never a reason to study fingerprinting codes that resist adversarial corruptions as is needed here. We construct such codes next.

4.2 Constructing robust fingerprinting codes

To trace high error-rate pirate decoders we construct a δ-robust fingerprinting code for any $\delta \in [0, 1)$. Our construction is based on the fingerprinting codes of Boneh-Shaw [4]. We first briefly review their construction.

The Boneh-Shaw codes. The fully collusion resistant code for n users is built from the following set of n words Γ_0, where each word consists of $n+1$ blocks and each block is d-wide:

$$\begin{array}{lcccccc} & \underline{\text{block } 0} & & & \cdots & & \underline{\text{block } n} \\ \text{word 1:} & 0000 & 1111 & 1111 & & 1111 & 1111 \\ \text{word 2:} & 0000 & 0000 & 1111 & \cdots & 1111 & 1111 \\ \text{word 3:} & 0000 & 0000 & 0000 & & 1111 & 1111 \\ \vdots & \vdots & & \vdots & & \vdots & \\ \text{word } n\text{:} & 0000 & 0000 & 0000 & \cdots & 0000 & 1111 \end{array} \tag{6}$$

The total codeword length is $\ell = d(n+1)$. The code generator G picks a random permutation π on $(1, \ldots, \ell)$ and permutes the columns of Γ_0 according to π. It outputs the resulting n codewords as the code Γ with tracing-key $tk := \pi$.

Let W be a subset of codewords in Γ that does not include codeword number i. Observe that an adversary $\mathcal{A}$ who is given W, cannot distinguish columns belonging to block number $i-1$ from columns belonging to block number i. Therefore, one expects that the codeword $\bar{w}^*$ generated by $\mathcal{A}$ contains roughly the same number of '1's in block $i-1$ as in block i. In fact, if block i in $\bar{w}^*$ contains many more '1's that block $i-1$, then we can conclude that $\mathcal{A}$ can distinguish block i from $i-1$ and therefore $\mathcal{A}$ is in possession of codeword number i.

Suppose $\mathcal{A}$ is given words $W \subset \Gamma$ and let $\bar{w}^* \in F(W)$ be a codeword generated by $\mathcal{A}$. For $i = 0, \ldots, n$ let a_i be the weight of the ith block of $\bar{w}^*$, namely the number of 1s in block i. Computing the quantities $a_0, \ldots, a_n$ requires the tracing-key tk to undo the random permutation π. Boneh-Shaw show that if there is a gap between block i and $i-1$, namely

$$a_i - a_{i-1} > \Delta \qquad \text{where} \qquad \Delta := \sqrt{d \cdot \log_2(2n/\epsilon)} \tag{7}$$

then $\mathcal{A}$ is in possession of codeword number i with probability at least $1 - (\epsilon/n)$ (they actually prove a stronger statement, but that is not needed for our discussion).

Equation (7) gives a tracing algorithm that does not accuse an innocent i: output all $1 \le i \le n$ such that $a_i - a_{i-1} > \Delta$. However, we must ensure that there is always some i that satisfies (7). Since $\bar{w}^* \in F(W)$ we know that $a_0 = 0$ and $a_n = d$ and therefore there is some $1 \le i \le n$ for which $a_i - a_{i-1} > d/n$. If we ensure that $d/n > \Delta$ then equation (7) will be satisfied for some $1 \le i \le n$, as required. To ensure $d/n > \Delta$ we solve for d and obtain:

$$d \ge d_{\min} := 2n^2 \log_2(2n/\epsilon)$$

implying that the code length is $\ell = d_{\min} \cdot (n+1) = O(n^3 \log(n/\epsilon))$. Overall, we trace $\bar{w}^*$ to some codeword used to create it and our tracing algorithm never outputs an empty set.

Making Boneh-Shaw δ-robust. Suppose $\mathcal{A}$ is given words $W \subset \Gamma$ and let $\bar{w}^* \in F_?(W)$ be a codeword generated by $\mathcal{A}$ that contains at most $\delta \cdot \ell$ symbols '?'. For $i = 0, \ldots, n$ let b_i be the number of '?' symbols in block i of $\bar{w}^*$. Let a_i be the number of '1's in block i of $\bar{w}^*$.
We modify the original Boneh-Shaw tracing algorithm as follows.

step 1: use the tracing-key tk to compute a_i and b_i for all $i = 0, \ldots, n$;
step 2: output all $1 \leq i \leq n$ such that $a_{i+1} - a_i > \Delta$ or $b_{i+1} - b_i > \Delta$.

The same logic as in the original algorithm shows that if the tracing algorithm outputs i then codeword number i was used to create $\bar{w}^*$ with probability at least ϵ/n. To see why, we re-iterate that without codeword number i the adversary cannot distinguish columns in block i from columns in block $i-1$ and therefore cannot create a large gap between a_i and a_{i-1} or between b_i and b_{i-1}. Therefore, the existence of a gap indicates that codeword i was used to create $\bar{w}^*$.

It remains to ensure that the modified tracing algorithm will not output the empty set. The algorithm will output the empty set only if

$$a_{i+1} - a_i \leq \Delta \quad \text{and} \quad b_{i+1} - b_i \leq \Delta \qquad \text{for all } i = 1, \ldots, n \tag{8}$$

Moreover, we have the following facts:

- Since $\bar{w}^* \in \mathbb{F}_?(W)$ we know that $a_0 = 0$ (i.e. there can be no 1 in block 0) and $a_n + b_n = d$ (i.e. there can be no 0 in block n).
- Since $\bar{w}^*$ contains at most $\delta\ell$ symbols '?', there must be some block $0 \leq j \leq n$ such that $b_j \leq \delta\ell/(n+1)$. Since $\ell = (n+1)d$ we obtain $b_j \leq \delta d$.

Using (8) and $a_0 = 0$ we deduce that $a_n \leq \Delta n$. Using (8) and $b_j \leq \delta d$ we deduce that $b_n \leq \delta d + \Delta n$. Therefore, if (8) holds then it must be that:

$$d = a_n + b_n \leq 2\Delta n + \delta d = 2n\sqrt{d \cdot \log_2(2n/\epsilon)} + \delta d \tag{9}$$

If we choose d sufficiently large so that (9) is false then (8) cannot hold and the tracing algorithm will output a non-empty set. Solving for d we obtain

$$d_{\min} > \frac{4n^2}{(1-\delta)^2} \cdot \log(2n/\epsilon)$$

which leads to a code of length $\ell = O\big((n^3/(1-\delta)^2) \cdot \log(2n/\epsilon)\big)$.

We obtain a δ-robust fingerprinting code for any $\delta \in [0, 1)$. This in turn leads to a fully collusion-resistant traitor-tracing system with constant size ciphertext and private keys of size ℓ. The tracing algorithm works by constructing $\bar{w}^*$ using experiments $\mathsf{RobustTR}_j$ and then running the robust fingerprint tracing algorithm on $\bar{w}^*$.

Our δ-robust code generalizes to t-collusion resistance as in [4]. For t-collusion resistance the resulting code length is

$$\ell = O\big((t^4/(1-\delta)^2) \cdot \log n \log(2n/\epsilon)\big).$$

5 Conclusions

We constructed a t-collusion resistant traitor tracing system where ciphertext size is independent of n or t. The system makes use of advances in fingerprinting codes. For full collusion resistance one can take $t = n$, without increasing the ciphertext size. Although ciphertexts are short, private-key size is quadratic in t. Our tracing algorithm is blackbox and is based on repeated sampling of the pirate decoder. Tracing time requires about $O(t^2\lambda)$ interactions with the pirate decoder.

Our tracing algorithm can trace both perfect and imperfect pirate decoders. To trace decoders with error-rate δ we need to increase the size of the secret key to about $O(n^3\lambda/\delta)$. The ciphertext is still constant size. We trace high error-rate decoders using δ-robust fingerprinting codes which we construct by extending the Boneh-Shaw fingerprinting codes.

References

[1] O. Berkman, M. Parnas, and J. Sgall. Efficient dynamic traitor tracing. In *Proceedings of SODA '00*, 2000.

[2] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 338–353, London, UK, 1999. Springer-Verlag.

[3] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Eurocrypt '06*, 2006. Full version available at `http://eprint.iacr.org/2006/045`.

[4] Dan Boneh and James Shaw. Collusion secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998. Extended abstract in Crypto '95.

[5] Dan Boneh and Brent Waters. A fully collusion resistant broadcast trace and revoke system with public traceability. In *ACM Conference on Computer and Communication Security (CCS)*, 2006.

[6] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[7] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 257–270, London, UK, 1994. Springer-Verlag.

[8] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[9] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography - PKC 2003*, volume 2567 of *LNCS*, pages 100–115, 2003.

[10] Amos Fiat and T. Tassa. Dynamic traitor tracing. In *Proceedings of Crypto '99*, volume 1666 of *LNCS*, pages 354–371, 1999.

[11] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 372–387, London, UK, 1999. Springer-Verlag.

[12] M. T. Goodrich, J. Z. Sun, , and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto '04*, volume 2204 of *LNCS*, 2004.

[13] H. Guth and B. Pfitzmann. Error- and collusion-secure fingerprinting for digital data. In *Information Hiding '99*, volume 1768 of *LNCS*, pages 134–145, 1999.

[14] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of Crypto '02*, volume 2442 of *LNCS*, pages 47–60, 2002.

[15] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In *ACM Workshop in Digital Rights Management – DRM 2001*, pages 22–39, London, UK, 2001. Springer-Verlag.

[16] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In Joan Feigenbaum, editor, *ACM Workshop in Digital Rights Management – DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages pp. 32–50. Springer, 2002.

[17] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 450–465, London, UK, 2002. Springer-Verlag.

[18] K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Proceedings of Eurocrypt '98*, pages 145–157, 1998.

[19] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002.

[20] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 41–62, London, UK, 2001. Springer-Verlag.

[21] Moni Naor and Benny Pinkas. Threshold traitor tracing. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 502–517, London, UK, 1998. Springer-Verlag.

[22] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*, pages 1–20, London, UK, 2001. Springer-Verlag.

[23] B. Pfitzmann. Trials of traced traitors. In *Proceedings of Information Hiding Workshop*, pages 49–64, 1996.

[24] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *Proceedings of the ACM Conference on Computer and Communication Security*, pages 151–160, 1997.

[25] Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. In *Proceedings of Crypto '00*, volume 1880 of *LNCS*, pages 316–332, 2000.

[26] Alice Silverberg, Jessica Staddon, and Judy L. Walker. Efficient traitor tracing algorithms using list decoding. In *Proceedings of ASIACRYPT '01*, volume 2248 of *LNCS*, pages 175–192, 2001.

[27] Jessica N. Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. Cryptology ePrint 2000/004, 2000.

[28] D. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM Journal on Discrete Math*, 11(1):41–53, 1998.

[29] D. Stinson and R. Wei. Key preassigned traceability schemes for broadcast encryption. In *Proceedings of SAC '98*, volume 1556 of *LNCS*, 1998.

[30] Gabor Tardos. Optimal probabilistic fingerprint codes. In *Proceedings of STOC '03*, pages 116–125, 2003.

[31] V. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of 2003 DRM Workshop*, 2003.

[32] Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In *Proceedings CT-RSA '01*, volume 2020 of *LNCS*, pages 392–407, 2001.

A Definition of Tracing Traitors

Initially, we view a pirate decoder PD as a probabilistic circuit that takes as input a ciphertext C and outputs some message m or $\perp$. A Traitor-Tracing system, then, consists of the following four algorithms:

Setup(n, λ) The setup algorithm takes as input n, the number of users in the system, and the security parameter λ. The algorithm outputs a public key bk, a secret tracing key tk, and private keys $sk_1, \ldots, sk_n$, where sk_u is given to user u.

Encrypt(bk, m) Encrypts m using the public broadcasting key bk and outputs ciphertext C.

Decrypt(j, sk_j, C) Decrypt C using the private key sk_j of user j. The algorithm outputs a message m or $\perp$.

Trace$^{PD}(tk)$ The tracing algorithm is an oracle algorithm that is given as input the tracing key tk. The tracing algorithm queries the pirate decoder PD as a black-box oracle. It outputs a set S which is a subset of $\{1, 2, \ldots, n\}$.

All these algorithms must run in polynomial time in λ and n. Moreover, the system must satisfy the following **correctness property**:
for all $j \in \{1, \ldots, n\}$ and all messages m:

Let $\big(bk, tk, (sk_1, \ldots, sk_n)\big) \stackrel{\text{R}}{\leftarrow} \mathit{Setup}(n, \lambda)$ and $C \stackrel{\text{R}}{\leftarrow} \mathit{Encrypt}(bk, m)$.

Then $\mathit{Decrypt}(j, sk_j, C) = m$.

Security. We define security of the traitor tracing scheme in terms of the following two natural games, called message-hiding and traceability.

Game 1. The first game is the standard **Semantic Security Game**. It says that the system is semantically secure to an outsider who does not possess any of the private keys. Since this is a standard notion we do not give the game details here. We denote the advantage of adversary $\mathcal{A}$ in winning this game as MH Adv$[\mathcal{A}, TT(n)](\lambda)$.

Game 2. The second game captures the notion of **Traceability against t-collusion**. For a given n, t, λ, the game proceeds as follows (both challenger and adversary are given n, t, and λ as input):

1. The adversary $\mathcal{A}$ outputs a set $T = \{u_1, u_2, \ldots, u_j\} \subseteq \{1, \ldots, n\}$ of at most t colluding users.
2. The challenger runs *Setup*(n, λ) and provides *bk* and $sk_{u_1}, \ldots, sk_{u_j}$ to $\mathcal{A}$. It keeps *tk* to itself.
3. The adversary $\mathcal{A}$ outputs a pirate decoder *PD*.
4. The challenger now runs $\mathit{Trace}^{PD}(tk)$ to obtain a set $S \subseteq \{1, \ldots, n\}$. Note that *Trace* is only given black-box oracle access to *PD*.

We say that the adversary $\mathcal{A}$ wins the game if the following two conditions hold:

- The decoder *PD* is useful. That is, for a randomly chosen m in the finite message space, we have that
$$\Pr[PD(\mathit{Encrypt}(bk, m)) = m] = 1$$
- The set S is either empty, or is not a subset of T.

We denote by TR Adv$[\mathcal{A}, TT(n,t)](\lambda)$ the probability that adversary $\mathcal{A}$ wins this game.

Definition 2. *We say that a Traitor Tracing system TT is t-collusion resistant if for all $n > t$ and all polynomial time adversaries $\mathcal{A}$ we have that* MH Adv$[\mathcal{A}, TT(n)](\lambda)$ *and* TR Adv$[\mathcal{A}, TT(n,t)](\lambda)$ *are negligible functions of λ.*

In Game 2 we require the pirate decoder *PD* to be perfect and decrypt all well-formed ciphertexts. We discuss imperfect pirate decoders in Section 4. Also note that we are modeling a stateless (resettable) pirate decoder — the decoder is just an oracle and maintains no state between activations. Non stateless decoders were studied in [15].

Minimal access decoders. The black-box tracing model described above is often called the *full access model* — the tracer is given the decryptions output by *PD*. When the decoder *PD* is a tamper resistant box, such as a music player, the tracer does not get direct access to decryptions; it only sees whether a given ciphertext results in music being played or not. To address this issue we define a more restricted black-box tracing model called *minimal access tracing.* This model is similar to the game above with the exception that the challenger presents the tracing algorithm with a more restricted oracle $\mathcal{P}(\cdot,\cdot)$ which takes a message-ciphertext pair as input and outputs:

$$\mathcal{P}(m, c) = \begin{cases} 1 & \text{if } PD(c) = m \\ 0 & \text{otherwise} \end{cases}$$

We then modify Step 4 of Game 2 above so that challenger runs $Trace^{\mathcal{P}}(tk, \epsilon)$ to obtain a set $S \subseteq \{1, \ldots, n\}$. Consequently, in the minimal access game the tracing algorithm is given far more restricted access to PD. One can argue [2] that this model accurately captures the problem of tracing a stateless tamper resistant decoder. It is not difficult to see that our tracing algorithm works in the minimal access model.