Traitor Tracing with Constant Size Ciphertext

Dan Boneh[*] Stanford University
Moni Naor[†] Weizmann Institute of Science

August 15, 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 a 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 [16], 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 $sk_1, \ldots, sk_n$. The *Encrypt* algorithm encrypts the content using bk and the *Decrypt* algorithm decrypts using one of the sk_i.

[*]Supported by NSF and the Packard Foundation.

[†]Incumbent of the Judith Kleeman Professorial Chair. Supported by a grant from the Israel Science Foundation.

The tracing algorithm *Trace* is the most interesting — it takes *tk* as input and interacts with a 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 describe our system as a public-key scheme, namely *bk* is public and anyone who knows it can create broadcast messages (we could equally have described it as a secret key scheme). As in many traitor tracing constructions, the tracing key *tk* in our system must be kept secret. Our tracing algorithm is black-box: it need not look at the internals of the pirate decoder *PD* and only interacts with *PD* as if it were a decryption oracle.

A traitor tracing system is said to be t-collusion resistant if tracing succeeds 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 basic system provides the following parameters as a function of the total number of users n, collusion bound t, and security parameter λ:

$$\begin{aligned} \text{CT-len} &= O(\lambda) \\ \text{SK-length} &= O(t^2\lambda^2 \log n) \\ \text{Tracing-time} &= O(t^2\lambda \log n) \end{aligned}$$

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, 24, 33, 34, 11, 12, 8, 29, 1, 32, 30, 23], and algebraic, as in [20, 2, 25, 19, 9, 22, 36, 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 [25, 23, 15, 13, 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 [19] and others [6, 10] 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 system generates ciphertexts of constant *size*. It can provide constant rate (rate = 1) for long messages by using hybrid encryption (i.e. encrypt a short message-key using the traitor tracing system and encrypt 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 [26, 27, 37, 18, 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 [17] and others [21, 28, 31] show how to convert tracing systems for stateless decoders into tracing systems 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.

2 Collusion resistant fingerprinting codes

Since our construction is based on collusion resistant fingerprinting codes, we first review their definition [4]. Collusion resistant codes are designed for fingerprinting digital content, but are also used in traitor tracing systems (e.g. [7, 19, 10, 31]). 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 $\bar{w}_i = \bar{w}^{(j)}_i$. 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 $\mathrm{Adv}[(G(n, \epsilon), T, C), \mathcal{A}]$ be the probability that $\mathcal{A}$ wins the game.

	Boneh-Shaw [4]	Tardos [35]
Full collusion resistance	$\ell = O\big(n^3 \log(n/\epsilon)\big)$	$\ell = O\big(n^2 \log(n/\epsilon)\big)$
t-collusion resistance	$\ell = O\big(t^4 \log(n/\epsilon) \log(1/\epsilon)\big)$	$\ell = O\big(t^2 \log(n/\epsilon)\big)$

Table 1: Lengths of fingerprinting codes obtained by running $G(n, \epsilon)$

Definition 1. *We say that 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 $C \subseteq \{1, \ldots, n\}$, 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 $C \subseteq \{1, \ldots, n\}$ 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. Tardos [35] improved these results by constructing shorter codes. The resulting code lengths are summarized in Table 1. Throughout the paper, except for Section 4, we will primarily rely on the Tardos construction.

We note that Chor et al. [7] constructed collusion resistant codes, but 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 [33, 34, 12, 32, 30]

3 A traitor-tracing system with short ciphertexts

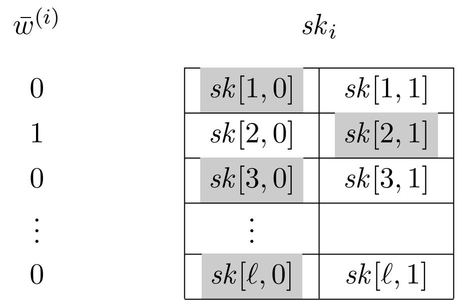
Let $\mathcal{E} := (G_{enc}, E_{enc}, D_{enc})$ be a public-key encryption system. We let $\mathcal{M}_\lambda$ denote the finite message space of $\mathcal{E}$ with security parameter λ. Throughout the paper we assume that its size $|\mathcal{M}_\lambda|$ is finite, but exponential in the security parameter. For simplicity we write $\mathcal{M}$ rather than $\mathcal{M}_\lambda$.

Let (G_{tt}, T_{tt}) be a fingerprinting code. Our traitor tracing system TT works as follows: (traitor tracing systems are defined in Appendix A)

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

1. Generate a fingerprinting code by running $(\Gamma, tk) \overset{\text{R}}{\leftarrow} G_{tt}(n, \epsilon)$.
   Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\} \subseteq \{0, 1\}^\ell$.
2. Generate 2ℓ public/secret key pairs by running G_{enc} 2ℓ times:

$$\text{for } i = 1, \ldots, \ell \text{ and } j = 0, 1 \text{ do:} \quad (pk[i, j],\ sk[i, j]) \overset{\text{R}}{\leftarrow} G_{enc}(\lambda)$$



$\bar{w}^{(i)}$	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: the key consists of the shaded boxes

3. For $i = 1,\ldots,n$ define $\quad sk_i \leftarrow \big(\ \bar{w}^{(i)},\ \ sk[1, w_1^{(i)}],\ \ldots,\ sk[\ell, w_\ell^{(i)}]\ \big)$.
   An example secret key is shown in Figure 1.
4. Define $\ bk \leftarrow \big(pk[1,0],\ pk[1,1],\ldots,pk[\ell,0],\ pk[\ell,1]\big)$
5. Output bk, tk, and $(sk_1,\ldots,sk_n)$

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

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

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

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

Including the index j in the ciphertext is done for convenience. In principle, j can be removed at the cost of forcing the decryptor to try all $1 \le j \le n$ until a j is found for which decryption succeeds (assuming the encryption system $\mathcal{E}$ embeds an integrity tag in ciphertexts). Clearly this is undesirable in all but extreme cases where the cost of bandwidth is much higher than the cost of computation.

The tracing algorithm: intuition

Suppose the adversary obtains a set of t secret keys and uses them to build a pirate decoder *PD*. For now let us assume that *PD* is a *perfect* decoder, namely it correctly decrypts well-formed ciphertexts. The t keys at the adversary's disposal correspond 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

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

Here m is some message not equal to 0. This ciphertext is invalid since the message encrypted under $pk[j,0]$ is different from the message encrypted under $pk[j,1]$. 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 perfect 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:

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

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

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

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 $Trace^{PD}(tk)$ works as follows:

1. For each j in $\{1, \ldots, \ell\}$ 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 $\text{Adv}[\mathcal{B}, \mathcal{E}]$ 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 polynomial time 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 (namely the bound on MH $\text{Adv}[\mathcal{A}, TT(n)]$ defined in Appendix A, Game 1) is immediate. We bound the adversary's advantage in winning the tracing game, namely TR $\text{Adv}[\mathcal{A}, TT(n,t)]$ 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*

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

Proof. Consider a modified tracing algorithm that produces a word $\bar{q}^*(\mathcal{A})$ as follows. For all $j = 1, \ldots, \ell$ run experiment TR'_j defined in Figure 2 and define $q_j \in \{0,1\}$ as:

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

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 $\Pr[w_j \ne q_j]$ must be negligible at all *critical* positions. In particular, for all critical positions $j \in \{1, \ldots, \ell\}$ there is a polynomial time semantic security adversary $\mathcal{B}$ for $\mathcal{E}$ such that

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

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 $\Pr[w_j \ne q_j]$ 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.

```
$m \xleftarrow{R} \mathcal{M}$
if all words in $C$ have a 1 in position $j$ do:
        $c_0 \xleftarrow{R} E_{enc}(pk[j,0],\ \boxed{0}),\qquad c_1 \xleftarrow{R} E_{enc}(pk[j,1],\ \boxed{0})$
else do:
        $c_0 \xleftarrow{R} E_{enc}(pk[j,0],\ \boxed{m}),\qquad c_1 \xleftarrow{R} E_{enc}(pk[j,1],\ \boxed{m})$
$\hat{c}^* \leftarrow (j, c_0, c_1)$
$\hat{m} \leftarrow PD(\hat{c}^*)$
```

Figure 2: Experiment TR'_j for $1 \leq j \leq \ell$

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 perfect 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. □

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. □

4 Tracing imperfect decoders

Our definition of secure traitor tracing in Appendix A requires that the adversary 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 (this may be useful for content that is repeated frequently). When the tracing algorithm from Section 3 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:

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

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

In this section we consider imperfect decoders. We assume that the broadcaster fixes an upper bound on δ and is not interested in tracing decoders with error-rate higher than δ since their usefulness is limited. Hence, we need only trace decoders PD whose error-rate is less than some fixed δ. Formally, the *Setup* algorithm for the traitor tracing system takes δ as a third input.

4.1 Robust fingerprinting codes

To address imperfect decoders we need a more sophisticated tracing algorithm as well as more powerful fingerprinting codes. We start with the requirements on the fingerprinting codes. When tracing an imperfect decoder PD there may be several coordinates where we fail to determine which keys are in the adversary's possession. Our traitor tracing algorithm (described in Section 4.3) will place a '?' in these coordinates. Consequently, unlike Section 3, the interaction with PD results in a "noisy" codeword

$$\bar{w}^* \in \{0, 1, ?\}^\ell .$$

PD can cause a '?' to appear in any coordinate in $\bar{w}^*$, as long as the overall number of '?' in $\bar{w}^*$ is bounded.

To trace noisy codewords we extend the definition of collusion resistant fingerprinting from Section 2. 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. That is, $\bar{w} \in \{0,1,?\}^\ell$ is **feasible** for W if for all $i = 1, \ldots, \ell$ either $\bar{w}_i = ?$ or there is a $j \in \{1, \ldots, t\}$ such that $\bar{w}_i = \bar{w}_i^{(j)}$. 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 Adv$[(G(n,\epsilon,\delta), T, C), \mathcal{A}]$ be the probability that $\mathcal{A}$ wins the game and we use this quantity in Definition 1 as follows.

Definition 2. *We say that a fingerprinting code (G,T) is **δ-robust 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*

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

*We say that (G,T) is **δ-robust 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*

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

Several results in the literature trace noisy codewords (codewords containing '?'). Most relevant is a recent result of Sirvent [31] who constructed codes satisfying Definition 2. Other results address much weaker notions of robustness, and are insufficient for our application:

- 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.

- Guth and Pfitzmann [14] 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.

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

4.2 Tracing using robust fingerprinting codes

We show that the traitor tracing system of Section 3 can be adapted to trace imperfect decoders, as long as the underlying fingerprinting code is robust. The system is unchanged except that the tracing algorithm works differently. It can trace a decoder PD with error rate at most δ as long as the underlying fingerprinting code is δ'-robust for a suitably chosen δ'.

The new tracing algorithm interacts with PD using the ℓ experiments defined in Figure 3. It runs all ℓ experiments and obtains ℓ pairs (p_j, q_j) for $j = 1,\dots,\ell$. To give some intuition we make two observations about the quantities p_j and q_j computed in these experiments:

- First, if $p_j > 0$ then the adversary $\mathcal{A}$ must possess $sk[j,0]$, since otherwise $\mathcal{E}$ is not semantically secure ($\mathcal{E}$ allows decryption of content without the key).

- Second, if p_j and q_j are "far apart" then $\mathcal{A}$ must possess $sk[j,1]$, since, again, otherwise $\mathcal{E}$ is not semantically secure ($\mathcal{E}$ distinguishes encryptions of 0 from encryptions of m).

Observe also that q_j is computed by running the pirate decoder on valid ciphertexts and therefore q_j is an estimate for the decoder's error rate for ciphertexts that use coordinate j.

These observations motivate defining the following codeword: for $j = 1,\dots,\ell$ let $w_j \in \{0,1\}$ be

$$w_j := \begin{cases} 0 & \text{if } p_j > 0 & (\mathcal{A} \text{ knows } sk[j,0]) \\ 1 & \text{if } p_j = 0 \text{ and } q_j > \frac{1}{\sqrt{\lambda}} & (\mathcal{A} \text{ knows } sk[j,1]) \\ \text{'?'} & \text{otherwise} & (p_j = 0 \text{ and } q_j \le \frac{1}{\sqrt{\lambda}}) \end{cases} \tag{6}$$

```
repeat the following steps $\lambda^2 \ln \ell$ times:
    $m \xleftarrow{R} \mathcal{M}$
    $c_0 \xleftarrow{R} E_{enc}\big(pk[j,0],\ \mathrm{m}\big)$,        $c_1 \xleftarrow{R} E_{enc}\big(pk[j,1],\ 0\big)$
    $c^* \leftarrow (j, c_0, c_1)$
    $\hat{m} \leftarrow PD(c^*)$
let $p_j$ be the fraction of times that $m = \hat{m}$

repeat the following steps $\lambda^2 \ln \ell$ times:
    $m \xleftarrow{R} \mathcal{M}$
    $c_0 \xleftarrow{R} E_{enc}\big(pk[j,0],\ \mathrm{m}\big)$,        $c_1 \xleftarrow{R} E_{enc}\big(pk[j,1],\ \mathrm{m}\big)$
    $c \leftarrow (j, c_0, c_1)$
    $\hat{m} \leftarrow PD(c)$
let $q_j$ be the fraction of times that $m = \hat{m}$
```

Figure 3: Experiment $\mathsf{RobustTR}_j$ for $1 \le j \le \ell$.

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

The codeword $\bar{w}^*$ satisfies two important properties: First, with high probability (i.e. probability at least $1 - f(\lambda)$ for some negligible function f), the fraction of '?' symbols in $\bar{w}^*$ is at most

$$\delta' := \delta/(1 - 2/\sqrt{\lambda}). \tag{7}$$

The proof is via a standard probabilistic argument showing that otherwise PD will incorrectly decrypt more than a δ fraction of well-formed ciphertexts: q_j is a good approximation to the actual probability $\hat{q}_j$ that when the jth coordinate is used PD decrypts correctly. If $\hat{q}_j \ge \frac{2}{\sqrt{\lambda}}$ then the probability that $q_j \le \frac{1}{\sqrt{\lambda}}$ is negligible. Now, if δ' is the fraction of locations where $\hat{q}_j < \frac{2}{\sqrt{\lambda}}$ then PD's error rate is at least $\delta'(1 - \frac{2}{\sqrt{\lambda}})$ which must be less than δ. Soving for δ' gives (7). (We assume that λ is sufficiently large to ensure that $\delta' < 1$.)

Second, one can show an analogue of Lemma 2. Let C be the set of codewords corresponding to the set of private keys in the adversary's possession. Then semantic security of $\mathcal{E}$ implies that $\bar{w}^*$ is contained in $F_?(C)$, with high probability. To see this we have to consider two cases: the probability that $p_j > 0$ when the adversary does not possess $sk[j,1]$, which is negligible, and the probability the adversary has of making $p_j = 0$ but $q_j > \frac{1}{\sqrt{\lambda}}$ when not possessing $sk[j,1]$. If we replace in the first step of $\mathsf{RobustTR}_j$ the encryption of 0 with m, then the adversary's chance of success is exponentially small. If now for the original $\mathsf{RobustTR}_j$ the probability of this event is not negligible, then we have a distinguisher for encryptions of a collection of random messages from encryptions of the all 0 collection, violating the semantic security of $\mathcal{E}$.

These two facts show that if the underlying fingerprinting code is δ'-robust then applying the fingerprinting tracing algorithm to the codeword $\bar{w}^*$ will identify a non-empty subset of keys in the adversary's possession, with high probability. In summary, the modified tracing algorithm works as follows:

1. For each j in $\{1,\ldots,\ell\}$ run experiment $\mathsf{RobustTR}_j$ and define the word $\bar{w}^*$ as in (6).
2. Output $T_{tt}(\bar{w}^*,\ tk)$.

We argued that tracing succeeds as long as the underlying fingerprinting code is δ'-robust.

4.3 Constructing robust fingerprinting codes

It remains to construct a δ-robust code to be used in the traitor tracing system of Section 3. We extend the Boneh-Shaw fingerprinting code [4] to make it δ-robust for any fixed $\delta \in [0,1)$. We note that Sirvent [31] recently presented a related construction. Constructing these codes is purely combinatorial and does not depend on complexity assumptions.

The Boneh-Shaw codes

We begin with a brief review of the Boneh-Shaw code. 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:

$$\begin{array}{lcccccc} & \underline{\text{block } 0} & & & \cdots & & \underline{\text{block } n} \\ \text{word 1:} & 0000 & 1111 & 1111 & & 1111 & 1111 \\ \text{word 2:} & 0000 & 0000 & 1111 & \cdots & 1111 & 1111 \\ \text{word 3:} & 0000 & 0000 & 0000 & & 1111 & 1111 \\ \vdots & \vdots & & \vdots & & \vdots & \\ \text{word } n\text{:} & 0000 & 0000 & 0000 & \cdots & 0000 & 1111 \end{array} \tag{8}$$

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 $tk := \pi$.

For notational convenience we will occasionally ignore the permutation π and use the term "block i of Γ" or "block i of a codeword w" to mean the set of coordinates that correspond (via π^{-1}) to block number i in Γ_0.

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 from 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 than 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 and Shaw show that if there is a gap between block i and $i-1$, namely

$$a_i - a_{i-1} > \Delta \qquad \text{where} \qquad \Delta := \sqrt{d \cdot \log_2(2n/\epsilon)} \tag{9}$$

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 (9) gives a tracing algorithm that accuses an innocent i with probability at most ϵ: 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 (9). 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 (9) 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 $\quad \ell = d_{\min} \cdot (n+1) = O(n^3 \log(n/\epsilon))$.
Overall, we trace $\bar{w}^*$ to some codeword used to create it and our tracing algorithm never outputs an empty set.

A δ-robust variant of Boneh-Shaw

We show that to make the code δ-robust it suffices to increase the block width $d_{\min}$ to

$$d_{\min} := \frac{4n^2}{(1-\delta)^2} \cdot \log_2(2n/\epsilon)$$

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 $1 \le i \le n$ such that $\qquad a_{i+1} - a_i > \Delta \quad \text{or} \quad b_{i+1} - b_i > \Delta$.

The same logic as in the original algorithm shows that this tracing algorithm accuses an innocent party with probability at most 2ϵ. 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 $\bar{w}^*$.

More precisely, we argue that if user i is not a member of W then

$$\Pr[a_{i+1} - a_i > \Delta \quad \text{or} \quad b_{i+1} - b_i > \Delta] < 2\epsilon/n$$

By the union bound it suffices to show that

$$\Pr[a_{i+1} - a_i > \Delta] < \epsilon/n \quad \text{and} \tag{10}$$

$$\Pr[b_{i+1} - b_i > \Delta] < \epsilon/n \tag{11}$$

Let $A := a_{i+1} + a_i$ and $B := b_{i+1} + b_i$. Since the columns in blocks $i-1$ and i all look the same to $\mathcal{A}$ we can bound the quantity $\Pr[a_{i+1} - a_i > \Delta]$ using the following balls and bins experiment: the adversary throws A blue balls (corresponding to '1' symbols) and B red balls (corresponding to '?' symbols) at random into $2d$ bins, with one ball per bin. Let x_i be the random variable indicating the number of blue balls in the left d bins and x_{i+1} be the number of blue balls in the right d bins. For any $0 \le k \le d$ we have

$$\Pr[x_i = k \text{ and } x_{i+1} = A - k] = \frac{\binom{d}{k} \cdot \binom{d}{A-k} \cdot \binom{2d-A}{B}}{\binom{2d}{A} \cdot \binom{2d-A}{B}} = \frac{\binom{d}{k} \cdot \binom{d}{A-k}}{\binom{2d}{A}}$$

This hyper-geometric distribution is the one analyzed in [4, Lemma 5.2] where they showed that

$$\Pr[x_{i+1} - x_i > \Delta] < \epsilon/n$$

which gives us the bound (10). The bound (11) follows similarly.

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

$$a_{i+1} - a_i \leq \Delta \quad \text{and} \quad b_{i+1} - b_i \leq \Delta \qquad \text{for all } i = 1, \ldots, n \tag{12}$$

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 (12) and $a_0 = 0$ we deduce that $a_n \leq \Delta \cdot n$. Using (12) and $b_j \leq \delta d$ we deduce that $b_n \leq \delta d + \Delta \cdot n$. Therefore, if (12) holds then it must be that:

$$d = a_n + b_n \leq 2\Delta n + \delta d = 2n\sqrt{d \cdot \log_2(2n/\epsilon)} + \delta d \tag{13}$$

If we choose d sufficiently large so that (13) is false then (12) 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

$$\ell = d_{\min} \cdot (n+1) = O\big((n^3/(1-\delta)^2) \cdot \log(2n/\epsilon)\big).$$

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 $\bar{w}^*$.

Sirvent [31, Section 4.3] shows that the method in [4] can be used to extend a δ-robust fully collusion resistant code to a δ'-robust t-collusion resistant code (for some $\delta' < \delta$), where the length of the code grows with $t^4 \log n$ rather than with n^3. For small collusions this shrinks the code length which in turn shrinks the size of secret keys in our system.

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 a perfect decoder 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 less than δ in $[0,1)$ we need to increase the size of the secret key to about $O(n^3\lambda/(1-\delta)^2)$. The ciphertext is still constant size. Tracing is done using δ-robust fingerprinting codes which we construct from the Boneh-Shaw code. It is likely that the fingerprinting code of Tardos [35] can also be made robust, but we leave this for future work.

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'05*, 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] N. Fazio, A. Nicolosi, and D. Phan. Traitor tracing with optimal transmission rate. In *Proceedings of ISC'07*, volume 4779 of *LNCS*, pages 71–88. springer, 2007.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] Jon Johansen. DeCSS, 1999. `http://en.wikipedia.org/wiki/DeCSS`.

[17] 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.

[18] 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.

[19] 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.

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

[21] T. Matsushita and H. Imai. A public-key black-box traitor tracing scheme with sublinear ciphertext size against self-defensive pirates. In *Proc. Asiacrypt'04*, pages 260–275, 2004.

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

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] D. Phan. Traitor tracing for stateful pirate decoders with constant ciphertext rate. In *Proceedings of Vietcrypt'06*, pages 354–365, 2006.

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

[30] 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.

[31] T. Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. In *Workshop on Coding and Cryptography*, 2007. Available at `http://eprint.iacr.org/2006/383.pdf`.

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

[33] 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.

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

[35] Gabor Tardos. Optimal probabilistic fingerprint codes. *Journal of the ACM*, 55(2), 2008. Extended abstract in STOC '2003.

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

[37] 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 Tracing Traitors: definition

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

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 message hiding 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 $\mathit{Setup}(n, \lambda)$ 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 perfect. That is, for a randomly chosen m in the finite message space, we have that
$$\Pr[PD(\mathit{Encrypt}(bk, m)) = m] = 1 \tag{14}$$
- 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 3. *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. Definition 3 easily generalizes to handle non-perfect decoders as in [3]: simply change (14) to

$$\Pr[PD(\mathit{Encrypt}(bk, m)) = m] \geq 1 - \delta$$

for some pre-agreed $\delta \in [0, 1)$ given to both the challenger and the adversary.

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 [17].

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 $\mathit{Trace}^{\mathcal{P}}(tk, \epsilon)$ 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.