# On the Security of Fully Collusion Resistant Traitor Tracing Schemes

Yongdong Wu [1] and Robert H. Deng[2]

[1]Institute for Infocomm Research, Singapore
[2] Singapore Management University, Singapore
wydong@i2r.a-star.edu.sg

**Abstract.** This paper investigates the security of FTT (fully collusion resistant traitor tracing) schemes in terms of DOT (Denial Of Tracing) and framing. With DOT attack, a decoder is able to detect tracing activity, and then prolongs the tracing process such that the tracer is unable to complete tracing job in a realistic time duration and hence has to abort his effort. On the other hand, by merely embedding several bytes of non-volatile memory in the decoder, we demonstrate, for the FTT schemes, how the decoder can frame innocent users at will. Furthermore, we propose a countermeasure on the framing attack.

## 1 Introduction

CASBAA (Cable and Satellite Broadcasting Association of Asia), which studied TV markets across 11 Asian countries, reported the number of illegal connections is expected to have risen 20% to $5.2 million in 2006, and pay-TV piracy in Asia is estimated to cost the industry $1.13 billion in 2006, up 6.6% from 2005 [1]. It is apparent that the need for content protection in the existing broadcast encryption systems is urgent and challenging. In a typical broadcast encryption system, each authorized user has a legal decoder embedded with a unique decryption key. A content distributor encrypts broadcast content such that only authorized users can decode protected content with their legal decoders. However, rogue authorized users, called traitors, may violate copyright protection policies by reverse-engineering the legal decoders (*e.g.*, [2]), sharing their decryption keys, constructing and distributing pirate decoders - illegal devices which are not registered with a broadcaster but are able to decrypt protected content. The broadcast encryption (BE) and traitor tracing (TT) model, an extension of the broadcast encryption model, incorporates various measures to deter authorized users from leaking their decryption keys.

BETT takes a confiscated pirate decoder, feeds it with special tracing messages, and outputs one or more identifiers of the colluders. Traitor tracing can be further divided into black-box, white-box and gray-box methods. Black-box tracing (*e.g.*, [3]-[6]) assumes that the tracer know nothing about the internals of the decoder. A specific black-box traitor tracing technique is black-box confirmation (*e.g.*, [7]) which is used to confirm whether or not a subset of users are

traitors. White-box tracing (*e.g.* [8]) assumes that the tracer know the internal detail (*e.g.*, the decryption key) of the pirate decoder. Gray-box tracing (*e.g.*, [9]) assumes that the tracer have partial knowledge on the internals of the decoder (*e.g.*, the decryption key for the protected content). The main challenge in the white-box and gray-box methods is reverse-engineering the decoder which is beyond the scope of the paper.

Most traitor tracing schemes in the literature are only effective when the number of traitors is much smaller than the number of authorized users. The notion of fully collusion resistant traitor tracing (FTT) was recently put forward in [10] [11]. A FTT scheme effectively identifies all traitors even if all authorized users collude in producing a pirate decoder.

In a private key FTT scheme, the tracer must know a private tracing key and therefore is assumed to be trusted by all. Boneh, Sahai and Waters [10] introduce a cryptographic primitive called Private Linear Broadcast Encryption (PLBE) and show that any PLBE gives a private key FTT scheme, refereed to as FTT1 scheme here. However, private key traitor tracing schemes suffer from the `customer's right problem`[1] [13]. A public key FTT scheme solves the above problem and allows anyone to run the tracing algorithm. Boneh and Waters [11] present a primitive called Augmented Broadcast Encryption (ABE) that is sufficient to construct public key FTT schemes, called FTT2 schemes in this paper. Both FTT1 and FTT2 schemes are resistant to an arbitrary number of colluders and are secure against adaptive stateless adversaries.

### 1.1 Features of BETT Systems

A BETT system consists of two algorithms: Broadcast Encryption (BE) and Traitor Tracing (TT). In the BE algorithm, a broadcaster encrypts a message $M$ into a ciphertext $C$, then transmits $C$ and some decryption tokens (or enabling blocks) over a broadcast channel. A user can correctly decrypt/decode $C$ if she possesses a legal decoder. On the other hand, the TT algorithm identifies traitors from a confiscated pirate decoder. A BETT system should have the following features:

(a) `Supporting large user population`: Broadcast (*e.g.*, pay-TV and encrypted satellite radio broadcast) delivers the same message to all authorized users simultaneously over a single broadcast channel, but incurs much more overhead than unicast communications in terms of system setting up and user device management. Hence, the larger the user population is, the more efficient utilization of the broadcast channel. Most broadcast applications are characterized by a large number $N$ of users (*e.g.*, $N \geq 10^6$ in [3][4]).

(b) `Accessibility of pirate decoders`: In order to start a tracing process, the tracer has to confiscate/access at least one pirate decoder. This prerequisition can be met when pirate decoders are easily accessible.

[1] In a private key traitor tracing scheme, since the broadcaster knows all the decryption keys, he may create a pirate decoder to frame any users at will.

(c) `Guaranteed QoS`: QoS (Quality of Service) is important in broadcast. For example, video stream must be of high quality; otherwise, users won't subscribe and pay for the service. This implies that a broadcaster can only insert in the broadcast channel a small amount of control data (*e.g.*, messages for tracing traitors) or noise.
(d) `Structured messages`: Decoders are usually designed to process messages formatted based on international or industrial standards. For example, content sent to VCD decoders follows MPEG format. Thus, messages transmitted in either normal broadcast mode or traitor tracing mode must have clearly specified format or structure; unstructured messages may crash legal decoders and/or alert pirate decoders that traitor tracing is underway. For this reason, traitor tracing schemes in [5] seem not practical since they assume that a pirate decoder is not able to distinguish random data from broadcast content.
(e) `Realistic tracing cost`: A practical traitor tracing scheme must allow a tracer to find traitors at an affordable cost either in terms of time (*e.g.*, within the life-time of the broadcast system) or in terms of budget (*e.g.*, at a small fraction of the operating cost of the broadcast system). It is simply not economically viable if a tracer has to spend 10 years or $10 millions in trying to identify a traitor. For this reason, we believe it is inappropriate to evaluate tracing cost based on notions in computational complexity.

For ease of exposition, notations used in this paper are listed in Table 1.

### 1.2 Our Contribution

The paper presents a DOT concept and attacks to FTT1 scheme [10] and FTT2 scheme [11]. Specifically,

(a) For tracing algorithm which is valid for stateless decoders, we introduce a novel Denial-Of-Tracing (DOT) mechanism which forces the tracing process taking too long time to complete such that the tracer is left with no choice but giving up the arm-race game.
(b) Assuming our pirate decoder is embedded with just several bytes of non-volatile memory, we demonstrate how the decoder is able to mislead the tracer to frame innocent users at will in both FTT1 and FTT2. We also improve FTT1 and FTT2 to prevent this framing attack.

The rest of the paper is organized as follows. Section 2 first briefly describes the DOT attack for pirate decoder. Sections 3 and 4 illustrate how to customize our pirate decoder to exploit the weaknesses in FFT1 and FFT2 respectively. Section 5 presents a stateful version of the decoder, and shows how it leads a tracer in FTT1 and FTT2 to frame innocent users. Section 6 draws our conclusion.

**Table 1.** Notations and abbreviations

| | |
|---|---|
| $N$ | the number of users |
| $U_j$ | the $j^{\text{th}}$ user |
| $\mathcal{D}_j$ | decoder of the $j^{\text{th}}$ user |
| $n$ | the actual number of collusion traitors |
| $t$ | the maximum number of tolerable traitors, $n \leq t \leq N$ |
| $U_{\pi_j}$ | the $j^{\text{th}}$ traitor, $\pi_j < \pi_{j+1}$ |
| $\mathbb{T}$ | the set of the traitors |
| $\mathcal{D}$ | a pirate decoder |
| $M$ | plaintext message |
| $C$ | ciphertext used in broadcasting or tracing mode |
| $M_j$ | output of decoder $\mathcal{D}_j$ |
| $\tau$ | initialization time of the pirate decoder $\mathcal{D}$ |
| $\epsilon$ | the minimal successful decoding probability of a useful decoder |
| $\lambda$ | a security parameter |
| BETT | Broadcast Encryption and Traitor Tracing |
| DOT | Denial Of Tracing |
| FTT | Fully collusion resistant Traitor Tracing |
| FTT1 | private FTT [10] |
| FTT2 | public FTT [11] |

## 2 Denial-of-Tracing

To start the DOT attack, a pirate decoder should have a *RST* unit which resets the pirate decoder to an initialization state. Usually, *RST* is able to be activated (1) externally: a traitor tracing scheme designed for stateless decoders demands that a decoder be reset before each new trail begins in order to clear any state information recorded by the decoder; or (2) internally, the *RST* unit trigs itself whenever the decoder detects that tracing is underway. In either case, the *RST* unit introduces an initialization (or intentional) delay $\tau$ that is related to the toleration time for a user to start a decoder. Thus, given the number $\omega$ of trials, the total tracing time is

$$T_{trace} = \omega \times \tau.$$

Generally speaking, $T_{trace}$ is bounded by the tracing budget and/or the lifetime of the broadcast system. When the tracer confiscates several pirate decoders and traces them in parallel, $T_{trace}$ serves as a good estimation of the total tracing cost. Therefore, if the number of tracing trials is too large so that the tracing time surpasses the budget, the tracer will have to give up. Hence, if any traitor tracing algorithm for stateless decoder requires a lot of trials, it is vulnerable to DOT attack in nature. In the following, we will employ DOT attack to defeat FTT1 and FTT2 tracing methods.

## 3 DOT Attack to FTT1

FTT1 is the first fully collusion resistant traitor tracing scheme with ciphertext of size $O(\sqrt{N})$ and private key of size $O(1)$. In this section, we demonstrate how a pirate decoder with DOT attack invalidates FTT1.

### 3.1 Overview of FFT1

Boneh, Sahai and Waters propose a private linear broadcast encryption (PLBE) primitive, and employ PLBE to realize a traitor tracing algorithm $\texttt{Trace}(\cdot)$ in FTT1 [10].

**PLBE** PLBE consists of a set of cryptographic functions $\texttt{Setup}(\cdot)$, $\texttt{Encrypt}(\cdot)$, $\texttt{TrEncrypt}(\cdot)$, and $\texttt{Decrypt}(\cdot)$.

- $\texttt{Setup}(N, \lambda)$: a probabilistic algorithm that takes as input $N$ and a security parameter $\lambda$. The algorithm runs in polynomial time in $\lambda$ and outputs a public key $PK$, a secret key $TK$ and private keys $K_1, \ldots, K_N$, where $K_i$ is given to user $U_i$.
- $\texttt{Encrypt}(PK, M)$: takes as input $PK$ and a message $M$. It outputs a ciphertext $C$ which is broadcast to all users.
- $\texttt{TrEncrypt}(TK, i, M)$: takes as input $TK$, an integer $i$ satisfying $1 \leq i \leq N+1$, and a message $M$. It outputs a ciphertext $C$. $\texttt{TrEncrypt}(\cdot)$ is primarily used for traitor tracing. Note that $\texttt{TrEncryptL}(TK, 1, M)$ outputs a ciphertext distribution that is indistinguishable from the distribution generated by $\texttt{Encrypt}(PK, M)$.
- $\texttt{Decrypt}(j, K_j, C, PK)$: takes as input the private key $K_j$ for user $j$, a ciphertext $C$, and the public key $PK$. The algorithm outputs a message $M$ or $\perp$.

Loosely speaking, PLBE is similar to an encryption system whose secret keys are constructed with a hash chain. Formally, the *PLBE property* is as follows:

$\forall i \in [1, N+1]$ and $j \in [1, N]$, and message $M$:
Let $(PK, TK, (K_1, \ldots, K_N)) \leftarrow \texttt{Setup}(N, \lambda)$,
and $C \leftarrow \texttt{TrEncrypt}(TK, i, M)$.
If $j \geq i$ then $\texttt{Decrypt}(j, K_j, C, PK) = M$.

The scheme FTT1 defines index hiding (IH) game as follows: if the decryption key $K_i$ is unknown, then one cannot distinguish encryptions with index $i$ from encryptions with index $i+1$. The indistinguishability of IH game implicitly express that if the decryption key $K_i$ is unknown, then one cannot distinguish encryptions with index $i$ from encryptions with index $i+x$ for $x > 0$. Otherwise, it makes no sense.

**Tracing Algorithm** Given a pirate decoder $\mathcal{D}$, and a $\epsilon > 0$, the linear tracing algorithm $\texttt{Trace}(TK, \epsilon)$ is shown in Fig.1. In the tracing algorithm, a tracer scans all users sequentially and compares the successful decoding probabilities of users' decoders. If the decoding probability of a user's decoder is abnormal, that user is incriminated.

Let the traitor set $\mathbb{T} = \phi$,
(a) For $i = 1$ to $N + 1$
The target event $\mathsf{E}$ is that the decoder has key $K_i$. Let $\eta = 0$. Repeat the following trial $\Omega_3 = 8\lambda(N \log N/\epsilon)$ times:
- Select a message $M$ from the finite message space at random.
- Let $C \leftarrow \texttt{TrEncrypt}(TK, i, M)$.
- Call $\mathcal{D}$ on input $C$ to get the output $\tilde{M}$. If $\tilde{M} = M, \eta \leftarrow \eta + 1$.

Let $\hat{p}_i$ be the fraction of trial times that $\mathcal{D}$ decrypted the ciphertexts correctly, thus, $\hat{p}_i = \eta/\Omega_3$.
(b) If $\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)$, event $\mathsf{E}$ is regarded to really occur, and thus $U_i$ is inserted into $\mathbb{T}$.
(c) Output $\mathbb{T}$ as the set of guilty colluders.

**Fig. 1.** Linear traitor tracing algorithm in FTT1.

### 3.2 Attack to FTT1 Scheme

**Detection of Tracing** Let $n$ be the number of traitors who actually colluded in the construction of a pirate decoder. The pirate decoder divides users into $n + 1$ subsets,

$$\begin{aligned} G_0 &= [1, \pi_1], \\ G_j &= (\pi_j, \pi_{j+1}], j = 1, 2, \ldots, n-1 \\ G_n &= (\pi_{n-1}, N]. \end{aligned}$$

where $\pi_i$ is the identification number of the $i$th traitor, the tracing algorithm $\texttt{Trace}(TK, \epsilon)$ in Fig.1, the tracer generates a ciphertext $C = \texttt{TrEncrypt}(TK, i, M)$, and sends $C$ to the pirate decoder $\mathcal{D}$. Upon receiving $C$,

- The decoder $\mathcal{D}$ gets $n$ plaintexts $M_{\pi_j}$ generated by traitor's decoder $\mathcal{D}_{\pi_j}$, $j = 1, 2, \ldots, n$. With reference to Fig.2, the index $i$ of the user under examination must be

$$i \in \begin{cases} G_0 : M_{\pi_1} = M_{\pi_n} \\ G_j : \exists j \in [1, n-1], M_{\pi_j} \neq M_{\pi_{j+1}} = M_{\pi_{j+2}} \\ G_n : \forall j, k, j \neq k, M_{\pi_j} \neq M_{\pi_k}, \end{cases} \quad (1)$$

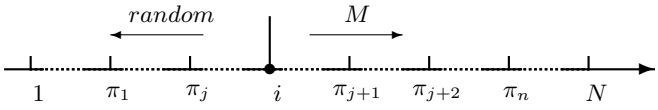


**Fig. 2.** Intervals a user index $i$ belongs to.

The above relationship reveals a lot of information about the index $i$ of user $U_i$. This invalidates the statement in [10] that "*a broadcast to users* $\{i, \ldots, N\}$ *should reveal no non-trivial information about* $i$". The decoder $\mathcal{D}$ translates Eq.(1) into the following

$$O_p = \begin{cases} 1 : i \in G_0 \\ -1 : i \in G_j, j \in [1, n-1] \\ 0 : i \in G_n \end{cases} \tag{2}$$

– If $O_p = 1$, the $ACT$ unit instructs the decoder to output the correct message $M$; however, if $O_p \neq 1$, the decoder knows it is being traced, and then takes protective actions against the tracer, e.g., DOT attack.

**DOT attack** FTT1 is designed to trace stateless decoder. Hence, a suspected decoder $\mathcal{D}$ is reset by the tracer before each trial, optionally, the decoder $\mathcal{D}$ can also be activated by the $RST$ unit if $O_p \neq 1$. Any of the resetting actions will start the reset process which will introduce a time delay $\tau$. Therefore, the total tracing time in FTT1 is at least

$$T_{trace} = \tau N \Omega_3 = 8\tau\lambda N^2 \log N/\epsilon$$

**Example 1**: Let $N = 100, \lambda = 100, \epsilon = 0.1$ (note $\epsilon = 0.01$ is used in [7]), $\tau = 1$ minute. Then a complete tracing process would take

$$\begin{aligned} T_{trace} &\approx 8 \times 1 \times 100 \times 100^2 \times \log 100/0.1 \\ &\approx 7.972 \times 10^7 \text{ minutes} \approx 150 \text{ years!} \end{aligned}$$

Even the binary search algorithm is used, the cost is still very high. Thus, if the decoder must be stateless by resetting after each tracing trial, the tracing algorithm in FTT1 is certainly infeasible in practice. However, if the decoder is kept stateful by not resetting it after each tracing trial, our pirate decoder can launch a framing attack (see Section 4).

## 4 DOT Attack to FTT2

The public fully collusion traitor tracing scheme (or FTT2) [11] allows anyone to trace a pirate decoder. In addition, a broadcaster can revoke any set of users. In this section we give a brief overview of FTT2 and point out its vulnerability against DOT attack.

### 4.1 Overview of FTT2 Scheme

The cryptographic primitive used to construct the traitor tracing algorithm $\texttt{Trace}(\cdot)$ in FFT2 is called augmented broadcast encryption (ABE) [11].

**ABE** ABE consists of a set of cryptographic functions: $\texttt{Setup}(\cdot)$, $\texttt{Encrypt}(\cdot)$, and $\texttt{Decrypt}(\cdot)$.

- $\texttt{Setup}(N, \lambda)$: A probabilistic algorithm that takes as input $N$, and a security parameter $\lambda$. The algorithm runs in polynomial time in $\lambda$ and outputs a public key $PK$ and private keys $K_1, \ldots, K_N$, where $K_j$ is given to user $U_j$.
- $\texttt{Encrypt}(S_{\mathcal{D}}, PK, i, M)$: Takes as input a subset of users $S_{\mathcal{D}} \subseteq \{1, \ldots, N\}$, a public key $PK$, an integer $i$ satisfying $1 \leq i \leq N+1$, and a message $M$. It outputs a ciphertext $C$. This algorithm encrypts a message to the set of users $S_{\mathcal{D}} \cap \{i, \ldots, N\}$.
- $\texttt{Decrypt}(S_{\mathcal{D}}, j, K_j, C, PK)$: Takes as input a subset $S_{\mathcal{D}} \subseteq \{1, \ldots, N\}$, the private key $K_j$ for user $j$, a ciphertext $C$, and the public key $PK$. The algorithm outputs a message $M$ or $\perp$.

*ABE property*: For all subsets $S_{\mathcal{D}} \subseteq \{1, \ldots, N\}$, $\forall i, j \in \{1, \ldots, N+1\}$ (where $j \leq N$), and all messages $M$:
Let $(PK, (K_1, \ldots, K_N)) \leftarrow \texttt{Setup}(N, \lambda)$,
and $C \leftarrow \texttt{Encrypt}(S_{\mathcal{D}}, PK, i, M)$.
If $j \in S_{\mathcal{D}}$ and $j \geq i$, $\texttt{Decrypt}(S_{\mathcal{D}}, j, K_j, C, PK)$ yields $M$, otherwise, $\perp$.

**Tracing Algorithm** Given a pirate decoder $\mathcal{D}$, a predetermined $\epsilon > 0$ and a set $S_{\mathcal{D}}$, the tracing algorithm $\texttt{Trace}(S_{\mathcal{D}}, PK, \epsilon)$ is shown in Fig.3.

**Tracing All Traitors** Like the tracing algorithm in FTT1, the above $\texttt{Trace}(S_{\mathcal{D}}, PK, \epsilon)$ in FTT2 may identify only one traitor. To identify all the traitors, FTT2 revokes the identified traitors and iteratively updates the set $S_{\mathcal{D}}$. Specifically, the $\texttt{TraceAll}(S_{\mathcal{D}}, PK, \epsilon)$ algorithm is shown in Fig.4.

### 4.2 Attack to FTT2

Since the tracing algorithms in FTT2 and FTT1 are similar, they are vulnerable to the same kind of attacks. To keep the paper compact, we will not present the details of DOT attack to FFT2 here. It should be pointed out that FTT2 is much more vulnerable to DOT attack than FTT1 since it requires more tracing trials.

Let the suspected user set $\mathbb{T} = \phi$.

(a) For $i = 1$ to $N + 1$ do
The target event $\mathsf{E}$ is that the decoder has key $K_i$. Let $\eta = 0$. Repeat the following trial $\Omega_4 = 8\lambda(N/\epsilon)^2$ times:
- Sample message $M$ from the finite message space at random.
- Let $C = \texttt{Encrypt}(S_{\mathcal{D}}, PK, i, M)$.
- Call $\mathcal{D}$ on input $C$ to get decoder output $\tilde{M}$. If $\tilde{M} = M$, $\eta \leftarrow \eta + 1$.

Let $\hat{p}_i$ be the fraction of trial times that $\mathcal{D}$ decrypted the ciphertexts correctly, i.e., $\hat{p}_i = \eta/\Omega_4$.

(b) If $\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)$, then event $\mathsf{E}$ is considered true, and $U_i$ is added to set $\mathbb{T}$.

(c) Output $\mathbb{T}$ as the set of guilty colluders.

**Fig. 3.** Traitor tracing algorithm in FTT2.

(a) Set $S_{\mathcal{D}}$ as the set of all of users.
(b) Find one traitor $u$ with $\texttt{Trace}(S_{\mathcal{D}}, PK, \epsilon)$;
(c) Let $\mathbb{T} \leftarrow \mathbb{T} \cup \{u\}$ and $S_{\mathcal{D}} \leftarrow S_{\mathcal{D}} \backslash \{u\}$;
(d) If $\texttt{Trace}(S_{\mathcal{D}}, PK, \epsilon)$ outputs a traitor, go to step (b). Otherwise, quit.

**Fig. 4.** Tracing all of the traitors in FTT2.

**Example 2**: Let's estimate the linear tracing time in FTT2 with the same parameters as in Example 1, i. e., $\tau = 1$ minute, $N = 100, \epsilon = 0.1, \lambda = 100$. The total tracing time is

$$\begin{aligned} T_{trace} &= N \times \Omega_4 \times \tau = N \times 8\lambda(N^2/\epsilon^2) = 8\lambda\tau N^3/\epsilon^2 \\ &= 800 \times 10^6/10^{-2} = 8 \times 10^{10} \text{ minutes} = 1.5 \times 10^5 \text{ years!} \end{aligned}$$

Thus, it requires on the average about $7.7 \times 10^4$ years to trace a traitor among a small group of $N = 100$ users. If the trace-revoke process $\texttt{TraceAll}(S_{\mathcal{D}}, PK, \epsilon)$ is used to trace all the traitors, it would take about $7.7 \times 10^6$ years.

## 5 Framing Attack to FTT

In this section, we first discuss the soundness of the assumption of stateless pirate decoders. We then introduce a framing attack to FTT assuming that the pirate decoder has several bytes of non-volatile memory. We also propose a countermeasure to defeat the framing attack. For simplicity, we address FTT1 only since it is straightforward to extend the attack and the countermeasure to FTT2.

### 5.1 Stateless vs. Stateful Decoder

Many traitor tracing schemes including FTT assume stateless pirate decoders. However, this stateless assumption seems too restrictive for several reasons.

(a) Memory is getting cheaper. Embedding a few bytes of non-volatile memory, which essentially cost nothing, makes a decoder stateful.
(b) Most decoders require non-volatile memory to store private data and to keep track of history information. For example, in order to decode MPEG video, a VCD decoder has to store recently decoded pictures in rendering a movie.
(c) Many authors assume that a traitor tracing scheme for stateless decoders can be converted into a tracing scheme for stateful decoders using the conversion algorithm introduced by Kiayas and Yung [21]. However, this conversion algorithm works under two assumptions. First, the algorithm is designed for partial collusion resistant traitor tracing schemes, not for FTT which assumes that all users may be involved in collusion. Second, the algorithm assumes that it is possible to identify at least one traitor from a copy which is generated from several watermarked copies. This assumption is still questionable:
    - It is proved in [22] that no traitor can be identified from a pirated copy if the number of the traitors is above a threshold given that the probability of implicating innocent is low;
    - Secure Digital Music Initiative (`http://www.sdmi.org/`) aimed to develop and standardize four audio watermarking techniques to protect digital music from illegal playing, storing, and distributing. Unfortunately, all four schemes were broken [23]. Additionally, many other watermark schemes have been published and then broken. Even the well-cited spread spectrum method [24] is being challenged [25].

Thus it is still very challenging to convert a tracing scheme for stateless decoders into one for stateful decoders. This is more so for FTT.

### 5.2 Framing Attack to Stateful Decoder

Assuming that our pirate decoder $\mathcal{D}$ is embedded with a non-volatile memory counter $l_C$ (*e. g.*, 8 bytes) to record the number of ciphertexts received. By exploiting the deterministic scanning pattern during traitor tracing in FTT1, Fig.5 shows how the decoder frames innocent users.

After completing the tracing/anti-tracing game in Fig.5, the tracer observes that $\forall \alpha = \pi_1 + 2d - 1 \leq \pi_n$ with some integer $d$, the pirate decoder $\mathcal{D}$ decoded all ciphertexts $C = \texttt{TrEncrypt}(TK, \alpha, M)$, but none of the ciphertexts $C = \texttt{TrEncrypt}(TK, \alpha + 1, M)$. Hence, the tracer concludes that

$$\hat{p}_\alpha = 1 \quad \text{but} \quad \hat{p}_{\alpha+1} = 0$$

which in turns implies that,

$$\hat{p}_\alpha - \hat{p}_{\alpha+1} = 1 \geq \epsilon/(4N)$$

Let $l_C = 0$.
For each ciphertext received, $\mathcal{D}$ proceeds as follows.

(a) The *WATCH* unit computes $O_p$ using Eq. (2). If $O_p \neq -1$, the $ACT$ unit outputs the average of all the $\{M_{\pi_j}\}_{j=1}^{n}$ and then receives the next ciphertext.

(b) If $O_p = -1$, the $ACT$ unit detects tracing is underway. It increments $l_C$ by 1 and sets $\gamma = \lfloor l_C/\Omega_3 \rfloor + 1$. Since each user is examined $\Omega_3$ times continuously, the decoder knows that the user $U_{\pi_1+\gamma}$ is being examined.

(c) If ($\gamma \mod 2$) = 1, the $ACT$ unit instructs the decoder to output $M_{\pi_n}$ which is identical to $M$; otherwise, the decoder outputs some random data.

**Fig. 5.** Framing attack to FTT1 with stateful pirate decoder.

Based on the FTT1 tracing algorithm in Fig.1, the tracer identifies the innocent user $U_\alpha$ as a traitor. Clearly, many users can be framed in this way.

**Example 3**: Assuming users $U_3$, $U_8$ and $U_9$ are traitors who create a pirate decoder $\mathcal{D}$. In the tracing algorithm in Fig.1, if $3 < i \leq 8$, the pirate decoder detects tracing is underway on account of $M_3 \neq M_8 = M_9$. Hence, during the tracing process, the output for $i = 4, 6$ is from $D_9$, while the output for $i = 5, 7$ is from $D_3$. The tracer concludes that

$$\hat{p}_4 = \hat{p}_6 = 1 \quad \text{but} \quad \hat{p}_5 = \hat{p}_7 = 0$$

or equivalently,

$$\hat{p}_4 - \hat{p}_5 = 1 > \epsilon/(4N)$$

and

$$\hat{p}_6 - \hat{p}_7 = 1 > \epsilon/(4N)$$

Thus, users $U_4$ and $U_6$ are identified as traitors wrongly.

### 5.3 Countermeasure on Framing Attack

In FTT, a tracer scans users following a known pattern (linear or binary), thus a stateful decoder knows whose decoder is being examined. Therefore, if the tracer randomly scans users during tracing, the pirate decoder may fail to frame innocent users. Such an improved tracing algorithm is depicted in Fig.6.

It should be noted that we use structured messages in tracing; otherwise the pirate decoder can defeat tracing by always outputting noise whenever it sees an unstructured message. With the random scan pattern, the probability of $\hat{p}_i - \hat{p}_{i+1}$ is small due to semantic security of either PLBE or ABE if $U_i$ is innocent. Therefore, no innocent users in any interval $(\pi_j, \pi_{j+1}]$ can be accused.

Let the traitor set $\mathbb{T} = \phi$.
(a) The algorithm repeats the following trial $N\Omega_3$ times:
  (i) Sample a ***structured*** message $M$, and select $i \in_R [1, N]$ ***randomly***.
  (ii) Let $C = \texttt{Encrypt}(S_{\mathcal{D}}, PK, i, M)$.
  (iii) Call $\mathcal{D}$ on input $C$, and compare the output of $\mathcal{D}$ to $M$.
(b) Let $\hat{p}_i$ be the fraction of times that $\mathcal{D}$ decrypted the ciphertexts correctly.
(c) If $\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)$, then add $i$ to set $\mathbb{T}$.

**Fig. 6.** Countermeasure to the framing attack.

## 6 Conclusion

One of the novel notions and techniques we introduced is Denial-Of-Tracing (DOT). DOT attack forces a traitor tracing process to take a very long time to complete such that the tracer is left with no choice but giving up its tracing attempt. To this end, the proposed pirate decoder injects a fixed amount of delay in each tracing trial by resetting the decoder. This reset mechanism is activated either by the tracer as required by most traitor tracing schemes for stateless decoders, or automatically by the decoder itself upon detection of an active tracing trial. Many traitor tracing schemes require a large number of trails and therefore are vulnerable to DOT attack. As concrete examples, we demonstrated successfully DOT attacks to black-box confirmation based schemes, combinatorial-key schemes, the private fully collusion resistant traitor tracing (FTT1) scheme and the public fully collusion resistant traitor tracing (FTT2) scheme. We presented several methods to counter DOT attack, but none of them seems effective.

We discussed the limitations and the impractical assumption on stateless pirate decoders. Assuming that the proposed pirate decoder is embedded with just several bytes of non-volatile memory, as another contribution of the paper, we showed how the decoder is able to mislead a tracer to frame innocent users at will in both FTT1 and FTT2. We also discussed ways to prevent this framing attack.

## References


1. CASBAA, "Pay-TV piracy on the rise in Asia: study," AsiaMedia Media News Dialy, `http://news.yahoo.com/s/afp/20061024/ennew_afp/asiatvindustrycrime_061024162529`, Oct 24, 2006
2. Joris Evers, "Breaking through Apple's FairPlay," `http://news.cnet.com/Breaking+through+Apples+FairPlay/2008-1025_3-6129420.html`
3. B. Chor, A. Fiat and M. Naor, "Tracing Traitors," Crypto'94, LNCS 839, pp. 257-270, 1994.

4. B. Chor, A. Fiat, M. Naor, and B. Pinkas, "Tracing Traitors," *IEEE Transactions on Information Theory*, 46(3):893-910, May, 2000.
5. A. Kiayias and M. Yung, "Traitor Tracing with Constant Transmission Rate," Eurocrypt '02, LNCS 2332, pp. 450-465, 2002.
6. Y. Dodis and N. Fazio, "Public Key Trace and Revoke Scheme Secure Against Adaptive Chosen Ciphertext Attack," PKC, pp.100-115, 2003.
7. Y. Dodis, N. Fazio, A. Kiayias, and M. Yung,"Scalable Public-Key Tracing and Revoking," PODC, pp.190-199, 2003.
8. A. Kiayias and M. Yung, "Breaking and Repairing Asymmetric Public-key Traitor Tracing," ACM DRM 2002, LNCS 2696, pp. 32-50, 2002.
9. J. McGregor, Y. L. Yin, and R. Lee, "A Traitor Tracing Scheme Based on RSA for Fast Decryption," ACNS 2005, LNCS 3531, pp. 56-74, 2005.
10. D. Boneh, A. Sahai, and B. Waters, "Fully Collusion Resistant Traitor Tracing With Short Ciphertexts and Private Keys," Eurocrypt '06, LNCS 4004, pp. 573-592, 2006.
11. D. Boneh and B. Waters, "A Collusion Resistant Broadcast, Trace and Revoke system," ACM CCS 2006, `http://eprint.iacr.org/2006/298`
12. D. Boneh, A. Sahai, and B. Waters, "Fully Collusion Resistant Traitor Tracing," `http://eprint.iacr.org/2006/045`, full paper of [10]
13. L. Qiao and K. Nahrstedt, "Watermarking Schemes and Protocols for Protecting Rightful Ownership and Customer's Rights," *J. Vis. Commun. Image Representation*, vol. 9, pp. 194-210, 1998.
14. D. Tonien, R. Safavi-Naini, "An Efficient Single-Key Pirates Tracing Scheme Using Cover-Free Families," ACNS, LNCS 3989, 82-97, 2006.
15. N. Alon, J. Bruck, J. Noar, M. Noar and R. Roth, "Construction of Asymptotically Good Low-Rate Error-Correcting Codes thorough Pseudo-Random Graphs," *IEEE Trans. on Information Theory*, vol. 38, pp.509-516, 1992.
16. B. Pfitzmann and M. Waidner, "Asymmetric Fingerprinting for Larger Collusions," ACM Conference on Computer and Communication Security, pp151-160, 1997.
17. E. Gafni, J.Staddon, and Y. L. Yin, "Efficient methods for integrating traceability and broadcast encryption," CRYPTO'99, pp. 372-387, 1999.
18. M. Naor and B. Pinkas, "Threshold traitor tracing," CRYPTO'98, pp. 502-517, 1998.
19. Y. Dodis and N. Fazio, "Public key broadcast encryption for stateless receivers," ACM Digital Rights Management Workshop, LNCS 2696, pp.61-80, 2002.
20. D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," Crypto'01, LNCS 2139, pp. 41-62, 2001.
21. Aggelos Kiayias and Moti Yung, "On Crafty Pirates and Foxy Tracers," ACM Workshop in Digital Rights Management -DRM 2001, Lecture Notes in Computer Science (LNCS) 2320, pp.22-39, 2002.
22. F. Ergun, J. Kilian and R. Kumar, "A Note on the Limits of Collusion-Resistant Watermarks," EUROCRYPT'99, LNCS 1592, pp. 140-149, 1999.
23. S. A. Craver, M. Wu, B. Liu, A. Stubblefield, B. Swartzlander, D. S. Wallach, D. Dean, and E. W. Felten, "Reading Between the Lines: Lessons from the SDMI Challenge," 10th USENIX Security Symposium, pp.353-363, 2001.
24. I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Trans. Image Processing*, 6(12):1673-1687, 1997.
25. T. Kanti Das and S. Maitra,"Cryptanalysis of Correlation-Based Watermarking Schemes Using Single Watermarked Copy," *IEEE Signal Processing Letters*, 11(4):446-449, 2004.