Efficient Traitor Tracing from Collusion Secure Codes

Olivier Billet[1] and Duong Hieu Phan[2]

[1] Orange Labs, Issy-les-Moulineaux, France
[2] Université Paris 8, Saint-Denis, France
olivier.billet@orange-ftgroup.com, hieu.phan@univ-paris8.fr

Abstract. In this paper, we describe a new traitor tracing scheme which relies on Tardos' collusion secure codes to achieve *constant size* ciphertexts. Our scheme is also equipped with a black-box tracing procedure against pirates that are allowed to decrypt with some (possibly high) error rate while keeping the decoders of the lowest possible size when using collusion secure codes, namely of size proportional to the length of Tardos' code.

1 Introduction

One common issue in digital content distribution is the problem of broadcasting data to several legitimate users in a secure way. Therefore, the broadcaster usually encrypts its data for the legitimate users. This is for example the case in pay-TV systems which allow to restrict access to the content to subscribers only, or when distributing digital media such as DVDs encrypted such that they can be used with compliant readers only [1]. In these scenarios and many others, the legitimate users rely on a decryption box containing the secrets that are necessary to obtain the digital content from the broadcasted information; this decryption box can be a tamper resistant device such as a smart card, a firmware for an electronic appliance, or a software on a personal computer. Tamper resistant devices are hard and expensive since they are designed to withstand a large range of attacks from side-channels attacks to invasive attacks. This raises the following issue: What if legitimate users are able to extract the secrets from their decryption box and redistribute them?

Traitor tracing is a well known cryptographic means to discourage such indelicate users (hereafter called traitors) from redistributing their secrets: It provides a way of embedding different secrets into each user's decryption box so that even if several traitors collude to produce a pirate decoder from their shared secrets, an authority is able to trace at least one of them. The efficiency of a traitor tracing scheme can be evaluated through several parameters: the maximum size c of tolerated coalitions, the size of the broadcasted ciphertext, and the size of the decoders. While it is obvious to design a traitor tracing scheme with a ciphertext size linear in the total number N of users, efficiently resisting collusions when traitors have full access to their decoders is not straightforward. Since its introduction by Chor, Fiat, and Naor [8], several techniques have been proposed. A first class of schemes that we might call combinatorial is based on carefully choosing some subset of a set of master keys to be put in each decryption box. By analyzing the keys found in a pirate decoder, it is possible to trace one of the traitors. The schemes [8, 20, 13, 9, 19] belong to this family. Another class

of schemes is the public key traitor tracing schemes first introduced in [17] by Kurosawa and Desmedt. To this family belong for instance [2, 21, 10, 18, 7, 5]. A third class of schemes relying on the use of collusion secure codes (and thus combining ideas from the two previous classes) has been introduced by Kiayias and Yung in [16]. The schemes [24, 11, 31, 4] belong to this class. Several of these works also provided additional features apart from the basic traitor tracing properties. It has been shown how to cope with decoders that decrypt correctly only with some (non-negligible) probability [20]. Tracing the traitors using black-box access only to the pirate decoders has been first proposed in [9]. The notion of public traceability has been proposed in [23, 7].

Our work, as for instance [16, 31, 11], is based on the use of collusion secure codes. These schemes enjoy many nice and desirable properties: they support black-box tracing and the ratio between the ciphertexts and the plaintexts is constant. However, since these schemes use collusion secure codes for both the ciphertext and the key used in the decoders, the size of the ciphertexts and decoders is quite large, namely $O(c^4 \log(N/\epsilon))$ for resisting coalitions of at most c traitors with probability $1 - \epsilon$. Another drawback of [16] comes from the use of an all-or-nothing transform (AONT [26]) to prevent deletion of keys from the pirate decoders as a way to escape the tracing procedure based on the underlying collusion secure code. This AONT renders the scheme quite rigid and prevents the reduction of the ciphertext's size since it requires to use *every* key from the decoder in order to decrypt a ciphertext, and thus slows down the decryption process. Safavi-Naini and Wang propose in [28] to use collusion secure codes that support random deletion in any position. In [31], Sirvent constructs new collusion secure codes which support deletion of a number of positions chosen by the adversary in addition to the usual properties: this results in a black-box tracing procedure which accommodates even more powerful pirates than [16, 28] and allows to remove the need for AONT. However, codewords from collusion secure codes supporting adversarial erasure have length $\Omega(c^4 \log(N/\epsilon))$ and the size of the ciphertexts and decoders remains large. In this paper, we propose a scheme based on Tardos' collusion secure code with *constant size ciphertexts* and thus resolve a first issue with code based traitor tracing schemes. The independent work [4] also proposes a scheme with constant size ciphertexts, but to be able to trace pirate decoders with non-negligible error rate δ, the size of the decoders is $\Omega(c^4/(1-\delta)^2 \log(N/\epsilon))$ and tracing is accordingly expensive. This large complexity comes from the fact that the authors in [4] built a collusion secure code with the strong property of resisting erasure. While this might lead to useful applications in settings other than traitor tracing, we show that such a strong collusion secure code is not required here: Our scheme takes advantage of the specific setting of traitor tracing where it is possible to distinguish between erased and unreadable positions. As a result our scheme can rely on Tardos' code and, in addition to bring constant size ciphertexts, also allows decoders of size $O(c^2 \log(N/\epsilon))$ even when considering pirate decoders with high error rates δ.

2 Tardos' collusion secure codes

Fingerprinting with collusion secure codes allows to uniquely identify a digital document among several copies of it by embedding a fingerprint (a codeword). Such an identification scheme must be resilient to collusions of traitors trying to remove their fingerprints so as to escape identification. Therefore, collusion secure codes share some properties with traitor

tracing; However, the main assumption here (called the marking assumption) is that the traitors from a coalition are only able to identify the positions where the digits from their respective codewords differ; Such positions are called *detectable positions*. This assumption especially makes sense when fingerprinting data: apart from the codewords, the documents are identical, and it is easy to uncover places where two copies of a document differ.

Among the first collusion secure codes are the identifiable parent property (IPP) codes introduced in [8]; However, these codes are defined over large alphabets and are resilient in a restricted attack model. The marking assumption and a way to construct randomized collusion secure codes has first been proposed by Boneh and Shaw in [6]; The length of the codewords is $O(N^3 \log(N/\epsilon))$ for fully-collusion resistant codes and $O(c^4 \log(N/\epsilon))$ for codes resisting coalitions of at most c traitors. Tardos later introduced a new construction in [33] and proved that the size of its codewords is optimal: a length of $O(c^2 \log(N/\epsilon))$ is enough to resist coalitions of at most c traitors [3]. This obviously gives fully-collusion secure codes of length $O(N^2 \log(N/\epsilon))$.

2.1 Tardos' construction

We now briefly describe the generation of a Tardos collusion secure code as proposed in [33]. We additionally describe the associated tracing procedure.

Code generation. In order to generate a code for N users that resists to c-collusions, set the length $\ell = 100c^2 \log(\frac{N}{\epsilon})$ where ϵ is the false-positive error probability (that is, the probability that an innocent user is accused) of the tracing algorithm and randomly draw a sequence of probabilities p_i as follows:

$$p_i = \sin^2(r_i), \qquad i \in [\![1, \ell]\!] \tag{1}$$

where r_i is randomly drawn from $[t, \pi/2 - t]$ and $0 < t < \pi/4$ is chosen so that $300\, c \sin^2 t = 1$.

Each binary codeword w of the code is then constructed by choosing its i-th digit to be either '1' or '0' according to the probability p_i, that is: $\Pr[w_i = 1] = p_i$.

Tracing procedure. The authority traces a subset of the traitors from a coalition (of at most c traitors) that has produced some binary word v by computing an accusation sum Z_w for each possible codeword w via:

$$Z_w = \sum_{i=1}^{\ell} v_i \cdot \left(\bar{w_i} \sqrt{\frac{1-p_i}{p_i}} + (\bar{w_i} - 1) \sqrt{\frac{p_i}{1-p_i}} \right) \ ,$$

where $\bar{w_i}$ is the bit w_i viewed as an integer. Then, users corresponding to codewords w such that $Z_w > 20\, c \log(\frac{N}{\epsilon})$ are declared as traitors. Tardos proves that the probability of false-negative alarms (that is, the probability that no traitor is found) is then $\epsilon^{c/4}$.

2.2 Note about the marking assumption

Here we make some basic remark that will be used later on in this paper. Think about Tardos' code as a matrix containing the codewords in its rows. We note that the columns in

[3] Update: For a concrete and formal evaluation of the length of the robust Tardos code, we refer to [22, 3], where it is shown that the length is proportional to $c^2 \log^2 c$. Consequently, we need to add a factor of $\log^2 c$ in our length evaluation in this article.

Tardos' code are all treated identically: they are generated the same way and contribute to the accusation sum following the same rule. Moreover, these columns have been generated *independently*. This very simple fact allows one to use codewords of bigger length, say, twice or four times the length of Tardos' original codewords (i.e. $L = 2\ell$ or $L = 4\ell$) and still allows to trace the traitors by using *any* subset of positions of size ℓ. We stress here that the traitors can make an educated choice of the subset of positions instead of choosing them randomly. However, even in this case, the resulting set of codeword remains a perfectly valid instance of Tardos' code.

This remark is motivated by the usual marking assumption for collusion secure codes. Indeed, the most commonly used marking assumption is that traitors are able to identify positions in their code words only where the digits differ: this fits a wide range of settings, such as watermarking of digital content. However, in the following, we additionally consider that some of the positions—regardless of the fact that they can be identified or not—might be deleted by the traitors, so that this position does not hold the original digit anymore. This issue motivated the introduction of AONT in [16] and the introduction of collusion secure codes resisting erasure in [31, 4]. However, our above remark shows that expanding a Tardos' code of length ℓ to a length $L = \frac{1}{\beta}\ell$ allows to cope with collusions of at most c traitors that are able to delete up to $(1-\beta)L$ digits from their pirate word, and then fall back to the *classical* marking assumption on the untouched subset of $\beta L = \ell$ positions.

3 Traitor tracing schemes from collusion secure codes

3.1 Construction

Building the decoders. The main idea to build the decoder is to use two different set of keys, viewed as a pair of tables denoted $T^{(0)}$ and $T^{(1)}$, each consisting of L randomly drawn n-bit elements and to use them to recover u random values $k_1, \ldots, k_u$ broadcasted (in an encrypted form) to the users in order to derive the corresponding session key SK from the HEADER for the data encapsulation mechanism: Obviously, the idea of using such a pair of tables is to allow the embedding of the identity of the user a in her personal decoder: if I^a is an L-bit string carrying the identity of user a, then we create a table T^a specific to user a by choosing as the i-th element $T^a[i]$ the key $T^{(I_i^a)}[i]$. (Here, I_i^a denotes the i-th bit of the bit string I^a.)

Coming back to the derivation of the session key SK from HEADER, since each decoder either embeds a key from $T^{(0)}$ or a key from $T^{(1)}$, the above-mentioned values k_i must be encrypted under both of these keys. The derivation of the session key is then performed from an HEADER $= (r_1, \ldots, r_u, z_1^{(0)}, z_1^{(1)}, \ldots, z_u^{(0)}, z_u^{(1)})$ where the r_i are u randomly chosen indices from $[\![1, L]\!]$ and the z_i are values obtained by encrypting u randomly chosen n-bit values $k_1, \ldots, k_u$ using an encryption scheme $\tilde{E}$ as follows:

$$\forall i \in [\![1, u]\!], \qquad z_i^{(0)} = \tilde{E}_{T^{(0)}[r_i]}(k_i) \quad \text{and} \quad z_i^{(1)} = \tilde{E}_{T^{(1)}[r_i]}(k_i) \ .$$

Note that the number u of elements of the table T entering in the derivation of the session key SK also depends on other system parameters in a way that is going to be discussed later on. A brief description of how these elements fit together into our proposal for an implementation of the function F_K to be used in our key encapsulation mechanism within each decoder is given in Figure 1.

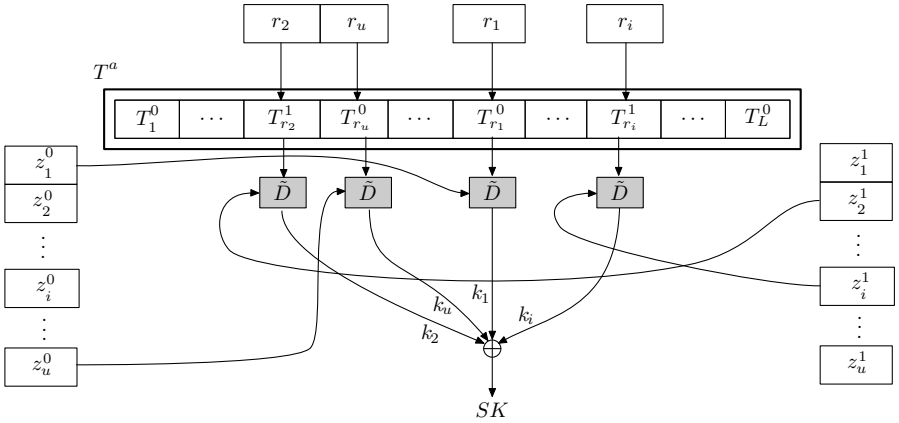

Fig. 1. The HEADER is made of the values r_i, $z_i^{(0)}$ and $z_i^{(1)}$ and the decoder of user a has access to table T^a. Using its bit string I^a, user a selects the correct values z_i to be decrypted: here, user a selects $z_1^{(0)}$, $z_2^{(1)}$, …, $z_i^{(1)}$, …, $z_u^{(0)}$. User a then decrypts these values with the corresponding keys $T^a[k_1]$, …, $T^a[k_u]$ from table T^a. The decrypted keys k_1, …, k_u are further combined together to form the session key $SK = k_1 \oplus \cdots \oplus k_u$.

Then, the data encapsulation mechanism is implemented as:

Encryption of M by broadcaster:

1. Draw $(r_1, r_2, \ldots, r_u)$ from $[\![1, L]\!]^u$ randomly;
2. Draw u elements k_1, k_2, …, k_u from $\{0,1\}^n$ randomly;
3. Encrypt the random values k_i as $z_i^{(0)} = \tilde{E}_{T^{(0)}[r_i]}(k_i)$ and $z_i^{(1)} = \tilde{E}_{T^{(1)}[r_i]}(k_i)$, for $i \in [\![1, u]\!]$;
4. Set HEADER $= (r_1, \ldots, r_u; z_1^{(0)}, z_1^{(1)}, z_2^{(0)}, z_2^{(1)}, \ldots, z_u^{(1)})$;
5. Derive the session key as $SK = k_1 \oplus k_2 \oplus \cdots \oplus k_u$;
6. Encrypt M with the underlying encryption algorithm E and the secret key SK: $C = E_{SK}(M)$;
7. Output the ciphertext $E'(M) = (\text{HEADER}, C)$.

Decryption of (HEADER, C) in decoders:

1. Extract r_1, …, r_u from HEADER;
2. Depending on the value I_i^a of the i-th bit of the user's identifying string I^a, compute

$$k_i = \tilde{D}_{T^a[r_i]}\left(z_i^{(I_i^a)}\right) \quad \forall i \in [\![1, u]\!] \ .$$

3. Derive the temporary secret key $SK = k_1 \oplus k_2 \oplus \cdots \oplus k_u$;
4. Use the underlying decryption algorithm to decrypt C with SK: $M = D_{SK}(C)$;
5. Output the plaintext $D'(C) = M$.

Pirate decoders. A pirate decoder is only required to decrypt valid ciphertexts with some probability τ. This is meant to take into account the case of coalitions of pirates dropping some of the secrets required to decrypt so as to help concealing their identity, which is highly critical in the setting of decoders based on collusion secure codes.

The tracing procedure. There are two main types of decoders: stateless decoders and stateful decoders. Stateless decoders do not record information between two decryption attempts whereas stateful decoders might memorize some information in order to help escaping the tracing procedure. We first describe a procedure against stateless decoders.

Our tracing procedure is derived from the general black-box tracing strategy for stateless decoders described in [9]: In order to decide if the decoder embeds the key $T^{(0)}[r]$ or the key $T^{(1)}[r]$ for some position r, the tracer provides the information to derive SK only for one of the key, that is, broadcasts $\tilde{E}_{T^{(0)}[r]}(k_1)$ and $\tilde{E}_{T^{(1)}[r]}(0)$; Therefore, if the decoder decrypts correctly this (invalid) ciphertext, the tracer deduces that the decoder knows $T^{(0)}[r]$.

Our tracing procedure also heavily relies on the property that the pirate decoders always embed at least ℓ digits that have been produced through the *classical* marking assumption (traitors can only put unreadable digits on detectable positions, the other positions are untouched) where ℓ is the required length for Tardos' code to be secure. We give sufficient conditions on the parameters u and β for this property to hold in Theorem 2 of the next paragraph.

We first describe the tracing procedure for $u = 1$. In this case, HEADER only consists of three values (r, z^0, z^1) where $\tilde{D}_{T^0[r]} = \tilde{D}_{T^1[r]}$. We call a 0-invalid header a header $(r, z^0, *)$ produced from a valid header (r, z^0, z^1) by replacing z^1 by a randomly chosen value '*' and similarly call $(r, *, z^1)$ a 1-invalid header. One can easily detect if the cell r of a decoder is coming from $T^{(0)}[r]$, $T^{(1)}[r]$ is unreadable or is wrong/erased; as noted previously, the ability to decide between unreadable cells (i.e. where the decoder knows the two possible keys) and the wrong/erased cells (i.e. where the decoder knows none of the keys) is fundamental to the tracing procedure. In order to distinguish these two types of positions, just follow the procedure:

- Input a number L/τ of valid headers with randomly chosen positions r (where τ is the decryption threshold of the decoder); every possible position therefore occurs τ^{-1} times on the average. Every position for which the decoder decrypted at least once, is declared an *inhabited* position. (Thus, inhabited position are the positions r for which the tracer knows for sure that the decoder embeds at least one of the values $T^{(0)}[r]$, $T^{(1)}[r]$.)
- For every inhabited position r, input a number τ^{-1} of 0/1-invalid headers. As soon as the decoder correctly decrypts a b-invalid header, deduce that cell r is coming from table $T^{(b)}$. If the decoder never decrypts b-invalid headers, deduce that the position r corresponds to a detectable position in the collusion secure code, that is, assume that the decoder knows both $T^{(0)}[r]$ and $T^{(1)}[r]$ and call this position an *unveiled* position.

(Note that the above procedure declares a position r to be '0', '1', or unveiled even though the pirate decoder refused to use the corresponding key $T^{(i)}[r]$ all of the time but once, i.e. used a probabilistic strategy to hide its choices.) Since the pirate decoder embeds at least ℓ digits (either from detectable positions or untouched from the traitors' original codewords) we are able to trace the traitors by applying Tardos' tracing procedure to these ℓ positions as explained in Section 2.

This procedure naturally extends to the case of $u > 1$. Remember that the pirate decoder must decrypt correctly with probability greater than the threshold τ the well formed headers. Therefore, for $\tau^{-1}L$ choices of the u-tuples, the tracer knows about every inhabited position (each time the decoder refuses to decrypt during this first phase no assumption is made, but when the decoder decrypts, u inhabited positions are learnt). Then, for every inhabited position r, the tracer considers 0/1-invalid headers corresponding to position r and chooses the remaining $u-1$ positions among the set of inhabited positions (discovered in the previous phase) randomly; the type ('0', '1', or unveiled) of position r is then determined as in the case $u = 1$ and the tracing procedure ends as before.

It is also possible to trace stateful decoders. Indeed, Kiayias and Yung proposed in [15] a generic strategy to convert a tracing procedure against stateless decoders into a tracing procedure against stateful decoders by using two versions of the plaintext watermarked differently. This strategy can be applied with a slight modification of our scheme: instead of encrypting k_1 under $T^{(0)}[r_1]$ and $T^{(1)}[r_1]$, the broadcaster encrypts k_1 under $T^{(0)}[r_1]$ and $\tilde{k}_1$ under $T^{(1)}[r_1]$; Then instead of $E_{SK}(M)$, the broadcaster encrypts the plaintext watermarked in two different ways M_1 and M_2 under $SK = k_1 \oplus k_2 \oplus \cdots \oplus k_u$ and under $\widetilde{SK} = \tilde{k}_1 \oplus k_2 \oplus \cdots \oplus k_u$ respectively, that is, provides $E_{SK}(M_1)$ and $E_{\widetilde{SK}}(M_2)$.

3.2 Security

In this paragraph we provide two results. The first one, given by Theorem 1, is that the encryption scheme we propose is secure. The second one is that our proposed implementation of the decoders is indeed resistant to coalitions of at most c-traitors and is given in Theorem 2.

Semantic security of a symmetric encryption scheme The semantic security of a symmetric encryption SKE = (KeyGen, Enc, Dec) is defined as follows:

Definition 1. *Let A be an adversary against SKE and λ be some security parameter. The adversary A chooses two messages, m_0 and m_1, of equal length, and gives them to an encryption oracle. The key generation $KeyGen(\lambda)$ generates a random key K, draws a random value $\sigma \in \{0,1\}$, and encrypts the corresponding message m_σ using the key K. The resulting ciphertext $c^\star = Enc_K(m_\sigma)$ is then provided to the adversary A. Finally, the adversary outputs $\hat{\sigma} \in \{0,1\}$. We define the advantage of A against SKE to be*

$$\mathsf{Adv}_A^{SKE}(\lambda) = \left| \Pr\left[\sigma = \hat{\sigma}\right] - \frac{1}{2} \right|$$

in the above attack game. We also define $\mathsf{Adv}^{SKE}(\lambda)$ as the maximum of all advantages $\mathsf{Adv}_A^{SKE}(\lambda)$ for all probabilistic, polynomial-time machines A. We say that SKE is semantically secure if $\mathsf{Adv}^{SKE}(\lambda)$ is negligibly for a security level λ.

Theorem 1. *Assume that the encryption schemes $(\tilde{E}, \tilde{D})$ and (E, D) are semantically secure. Let us assume that adversaries know for a fraction of at most 2α positions the corresponding entry from at least one of the two tables $T^{(0)}$ and $T^{(1)}$, and let $u(\lambda)$ be chosen so that $\pi_u(\lambda) = \binom{\alpha L}{u} / \binom{L}{u}$ is negligible for the security level λ. Then*

$$\mathsf{Adv}^{E'}(\lambda) \leq \pi_u(\lambda) + 2\mathsf{Adv}^{\tilde{E}}(\lambda) + \mathsf{Adv}^{E}(\lambda)$$

and thus (E', D') is semantically secure against the above adversaries.

Proof. First, note that for each choice of the tuple of indices $r_1, r_2, \ldots, r_u$, either the adversary knows at least one value in every of the u pairs $(T^{(0)}[r_1], T^{(1)}[r_1]), \ldots, (T^{(0)}[r_u], T^{(1)}[r_u])$, or she does not know $T^{(0)}[r_i]$ and $T^{(1)}[r_i]$ for at least one index r_i among $r_1, \ldots, r_u$. The first case happens at most $\pi_u = \binom{\alpha L}{u} / \binom{L}{u}$ of the times over the random choices of $r_1, \ldots, r_u$. In the other case, we can assume without loss of generality that the adversary A knows the $u-1$ remaining indices and (by renaming the indices) that the unknown values correspond to r_1, that is she does not know $T^{(0)}[r_1]$ nor $T^{(0)}[r_1]$. Let us note $s_0 = T^{(0)}[r_1]$ and $s_1 = T^{(1)}[r_1]$. As the keys to encrypt $k_2, \ldots, k_u$ are known to the adversary, the encryption of a message m of E' can be expressed, under the adversary's view, as $\big(\tilde{E}_{s_0}(k_1), \tilde{E}_{s_1}(k_1), k_2, \ldots k_u, E_{SK}(m)\big)$, where $SK = k_1 \oplus k_2 \oplus \cdots \oplus k_u$. Let $\kappa = k_2 \oplus \cdots \oplus k_u$ so that $SK = k_1 \oplus \kappa$.

We now wish to bound the advantage of the adversary in breaking (E', D'). To this end, let Game_0 be the original attack game played by the adversary A against (E', D'). We denote by $\psi = (e_0^\star, e_1^\star, k_2^\star, \ldots, k_u^\star, c^\star)$ the target ciphertext, we denote by σ the hidden bit generated by the encryption oracle, and we let $\hat{\sigma}$ be the bit outputted by A. Let T_0 be the event where $\sigma = \hat{\sigma}$. Also, let $k_1^\star$ denote the underlying message corresponding to the ciphertexts $e_0^\star$, $e_1^\star$ and $SK^\star$ denote the symmetric key used to encrypt m_σ, that is: $e_0^\star = \tilde{E}_{s_0}(k_1^\star)$, $e_1^\star = \tilde{E}_{s_1}(k_1^\star)$, and $c^\star = E_{SK}(m_\sigma)$.

We also define a modified game Game_1 which behaves just like game G_1, except that a completely random symmetric key SK^+ is used in place of the key $SK^\star$. Let T_1 be the event that $\sigma = \hat{\sigma}$ in this game Game_1.

It is straightforward to see that there is an oracle query machine A_1, whose running time is essentially the same as that of A, such that:

$$\big|\Pr[T_1] - \Pr[T_0]\big| \leq 2\mathsf{Adv}^{\tilde{E}}_{A_1}(\lambda) \ . \tag{2}$$

Indeed, the adversary A_1 just uses adversary A to play two independent games against $\tilde{E}$: one under the key s_0 and another under the key s_1. In the attack games that A_1 are playing against $\tilde{E}$, the challenged message k_1^x is equal to $SK^\star \oplus k_2^\star \oplus \cdots \oplus k_u^\star$ in game Game_0, and is equal to $SK^+ \oplus k_2^\star \oplus \cdots \oplus k_u^\star$ in game Game_1.

Finally, we observe that in this modified game Game_1, the key SK^+ is used to encrypt message m_σ and does not play any other role. Thus, in game Game_1, the adversary A is essentially carrying out an attack against E:

$$\left|\Pr[T_1] - \frac{1}{2}\right| \leq \mathsf{Adv}^{E}(\lambda) \ . \tag{3}$$

By combining Eq. (2) and Eq. (3) in the case where the adversary lacks at least one pair, we get:

$$\mathsf{Adv}^{E'}(\lambda) \leq \pi_u(\lambda) + (1 - \pi_u)\big(2\mathsf{Adv}^{\tilde{E}}(\lambda) + \mathsf{Adv}^{E}(\lambda)\big)$$

which proves the theorem. □

An immediate corollary of the previous theorem is that the encryption scheme prevents an attacker from dropping too many cells of its pirate decoder without dramatically dropping its probability of correctly decrypting. The following theorem in turn shows that this can be exploited to rely on the collusion secure code to trace at least one of the traitors.

Theorem 2. *Consider our construction for a traitor tracing scheme given in Sec. 3.1 with master tables $T^{(0)}$, $T^{(1)}$ of n-bit cells and length L, a number u of keys k_i, and where the identifying strings are taken from the c-collusion secure code for N users derived from Tardos' fingerprinting scheme such as explained in Sec. 2, that is of size $L = \frac{100}{\beta} c^2 \log\left(\frac{N}{\epsilon}\right)$. We claim that no coalition of less than c traitors can produce a pirate decoder with a decryption probability greater than 2^{-t} that can not be traced to at least one of the traitors as soon as β and u are chosen so that:*

$$\binom{\beta L}{u} \leq 2^{-t} \binom{L}{u} \ . \tag{4}$$

Proof. The idea of the proof is as follows: for the underlying fingerprint code to work, we need to ensure that at least (say) one fourth of the cells of the tables has to be kept. Forcing the pirate decoder to embed this number of cells to be able to decrypt correctly can be achieved by increasing the number u of required cells to derive the session key SK.

First of all, note that for a cell from the table T^P of the pirate decoder to be useful to the pirate, more than $n-t$ bits must be exact. (The remaining t bits can be guessed on the fly for a price of at most 2^{-t}, but more than t unknown bits would be too costly.) Therefore, either the pirate decoder stores $n-t$ bits or more of some cell (and thus the corresponding bit from the fingerprinting code can be deduced) or the decoder stores less than $n-t$ bits of the cell (and thus it is useless for the derivation of the session key).

From the above we deduce that we can assume that only a certain fraction $0 < \alpha \leq 1$ of the cells of the table are kept in the pirate decoder. Now the probability that the decoder is able to decrypt correctly is: $\pi_u = \binom{\alpha L}{u} / \binom{L}{u}$, so that the pirate decoder can not decrypt with probability higher than 2^{-t} by the hypothesis made at Eq. 4 if $\alpha < \beta$. Therefore the pirate decoder embeds more than βL digits and since the underlying Tardos' fingerprint code of length L has been expanded to $\frac{100}{\beta} c^2 \log\left(\frac{N}{\epsilon}\right)$, there remains $100c^2 \log\left(\frac{N}{\epsilon}\right)$ digits in the pirate word which allows Tardos' tracing algorithm to output a list of traitors as usual. □

3.3 Efficiency and sample parameters

We now propose a set of parameters for a sample implementation with the AES as the underlying encryption schemes (E, D) and $(\tilde{E}, \tilde{D})$. The key size is therefore chosen to be $n = 128$. For a number of users $N = 2^{30}$, setting $\beta = \frac{1}{2}$, and considering coalitions of at most $c = 100$ traitors, the expanded code has length $L \simeq 2^{24}$, and Theorem 2 gives the following data:

	$2^{-t} = \frac{10}{100}$	$2^{-t} = \frac{1}{100}$	$2^{-t} = \frac{1}{1000}$
u	4	7	10

4 Conclusion

The long series of work about traitor tracing schemes based on collusion secure codes shows that they can provide many interesting properties such as constant size ciphertexts, black-box tracing procedures against stateful (and possibly high error rate) pirate decoders. In contrast, the intriguing question of whether achieving trace and revoke capabilities is possible or not remains open.

References

1. AACS. `http://www.aacsla.com`.
2. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 1999.
3. Dan Boneh, Aggelos Kiayias, and Hart William Montgomery. Robust fingerprinting codes: a near optimal construction. In *Proceedings of the 10th ACM Workshop on Digital Rights Management, Chicago, Illinois, USA, October 4, 2010*, pages 3–12. ACM, 2010.
4. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertexts, February 2008. Manuscript available from `http://crypto.stanford.edu/~dabo/papers/`.
5. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer, 2006.
6. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465. Springer, 1995.
7. Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 542–558. Springer, 2005.
8. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.
9. Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.
10. Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 2002.
11. Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *Information Security – ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 71–88. Springer, 2007.
12. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1994.
13. Amos Fiat and Tamir Tassa. Dynamic traitor training. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 1999.
14. Aggelos Kiayias and Moti Yung. Self protecting pirates and black-box traitor tracing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 2001.
15. Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In Tomas Sander, editor, *Digital Rights Management Workshop*, volume 2320 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2002.
16. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 450–465. Springer, 2002.
17. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *Advances in Cryptology – EUROCRYPT '98*, pages 145–157, 1998.

18. Tatsuyuki Matsushita and Hideki Imai. A public-key black-box traitor tracing scheme with sublinear ciphertext size against self-defensive pirates. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2004.
19. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.
20. Moni Naor and Benny Pinkas. Threshold traitor tracing. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 1998.
21. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2001.
22. Koji Nuida. A general conversion method of fingerprint codes to (more) robust fingerprint codes against bit erasure. In *Information Theoretic Security, 4th International Conference, ICITS 2009, Shizuoka, Japan, December 3-6, 2009. Revised Selected Papers*, volume 5973 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2009.
23. Birgit Pfitzmann. Trials of traced traitors. In Ross J. Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 1996.
24. Duong Hieu Phan. Traitor tracing for stateful pirate decoders with constant ciphertext rate. In Phong Q. Nguyen, editor, *Progress in Cryptology – VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2006.
25. Duong Hieu Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming – ICALP 2006*, volume 4052 of *Lecture Notes in Computer Science*, pages 264–275. Springer, 2006.
26. Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *Fast Software Encryption – FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 1997.
27. Reihaneh Safavi-Naini and Yejing Wang. Collusion secure q-ary fingerprinting for perceptual content. In Tomas Sander, editor, *Digital Rights Management ACM CCS Workshop – DRM 2001*, volume 2320 of *Lecture Notes in Computer Science*, pages 57–75. Springer, 2002.
28. Reihaneh Safavi-Naini and Yejing Wang. Traitor tracing for shortened and corrupted fingerprints. In Joan Feigenbaum, editor, *Digital Rights Management ACM CCS Workshop – DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 81–100. Springer, 2003.
29. Alice Silverberg, Jessica Staddon, and Judy L. Walker. Efficient traitor tracing algorithms using list decoding. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2001.
30. Alice Silverberg, Jessica Staddon, and Judy L. Walker. Applications of list decoding to tracing traitors. *IEEE Transactions on Information Theory*, 49(5):1312–1318, 2003.
31. Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. In Daniel Augot, Nicolas Sendrier, and Jean-Pierre Tillich, editors, *Workshop on Coding and Cryptography – WCC '07*, pages 379–388, April 2007.
32. Douglas R. Stinson and Ruizhong Wei. Key preassigned traceability schemes for broadcast encryption. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography – SAC '98*, volume 1556 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 1999.
33. Gábor Tardos. Optimal probabilistic fingerprint codes. In *ACM Symposium on Theory of Computing – STOC 2003*, pages 116–125. ACM, 2003.