Traitor tracing scheme with constant ciphertext rate against powerful pirates

Thomas Sirvent

DGA/CELAR - IRMAR
thomas.sirvent@m4x.org

Abstract. Traitor tracing schemes are used to fight piracy when distributing securely some data to multiple authorized receivers: if some receivers collude and share their decryption keys to build some pirate decoder, a tracing procedure should be able to find at least one of these "traitors" from the pirate decoder. In this paper, we consider powerful pirate decoders, which may sometimes refuse to decrypt, or may try to detect when the tracing procedure is running. Most known traitor tracing schemes are not secure against this kind of pirate decoders: this is in particular the case of the schemes with constant ciphertext rate, which are the most efficient ones. We build then a new scheme, with constant ciphertext rate and security against powerful pirate decoders, using watermarking techniques. This scheme has the interesting feature that a receiver may decrypt the ciphertexts progressively, when it was not possible in previous schemes with constant ratio between ciphertext and plaintext.

Keywords: Traitor tracing, piracy, watermarking, collusion secure codes.

1 Introduction

Chor, Fiat and Naor present in [7] the first definition for traitor tracing schemes. Traitor tracing schemes are used when a supplier wants to send securely some data to multiple authorized users. But some authorized users, called traitors, may help unauthorized users to access this data. Such unauthorized users are called pirates.

To help the pirates, the traitors can simply obtain the protected data using their decryption keys and give this data to the pirates. A standard restriction to this behavior is that it may be difficult to transmit the protected data. The traitors can then proceed differently and give their decryption keys. With this knowledge, the pirates act exactly like authorized users, and easily obtain the data from the broadcasted ciphertext. Even worse, several traitors may collude and use the knowledge of all their decryption keys to build a pirate decoder.

A traitor tracing scheme deals with this threat: when a pirate decoder is caught, it can be used to find a traitor, i.e. an authorized user who participated in building the pirate decoder. This property is very pleasant, since it dissuades authorized users to reveal their keys: the supplier cannot prevent them from giving their keys, but it can trace them later, and incriminate them. This kind of schemes has direct applications to domains where only subscribers must receive the data: pay-television and on-line databases are good examples, where the data can moreover not be easily transmitted directly by the subscribers to pirates without risk.

1.1 Previous work

Different approaches have been used to build traitor tracing schemes. The first one is based on a well-chosen distribution of secret keys, suggested by Fiat and Naor in [11] for broadcast

encryption. These secret keys are used independently, and each user receives some of these secret keys. To decrypt a ciphertext, a pirate decoder needs a set of these secret keys: from the identification of the keys used by the pirate decoder, a tracer may find a traitor. The first traitor tracing schemes, proposed by Chor, Fiat, Naor and Pinkas in [7, 8, 20], use this design.

Public-key traitor tracing schemes have been later developed. Their idea is different: the decryption keys have a common property, which allows the decryption. Thus, a pirate decoder may build a new decryption key, from the keys given by the traitors. However, the use of a new decryption key may not hide the traitors. The first scheme with this kind of design has been proposed by Kurosawa and Desmedt in [17]. Other schemes have been suggested in the same way by Boneh and Franklin in [2], Mitsunari et al. in [19], To et al. in [24], Matsushita and Imai in [18], and Boneh et al. in [3, 5]. However, some of these schemes have been proved insecure, like the ones proposed in [17, 19, 24].

Kiayias and Yung suggest in [16] a third approach, which uses both previous techniques. They obtain public-key traitor tracing schemes for 2 users, and use several instances of this scheme, with a well-chosen distribution of the secret keys corresponding to the instances.

Different models for the pirate decoders have been studied. Almost all schemes mentioned previously deal with "black-box" pirate decoders: a tracer cannot directly obtain a decryption key or the description of a decryption mechanism used by the pirate decoder, it has to interact with the pirate decoder, sending decryption queries and analyzing the answers.

Kiayias and Yung present in [14, 15] pirate decoders with strong capabilities. In their model, a pirate decoder may be history-recording: the tracer cannot reset the pirate decoder to its initial state. In an other way, a pirate decoder may be abrupt: when it detects a tracing procedure running, it erases its decryption mechanism and tries to destroy the tracing mechanism, releasing for example a virus.

The ciphertext rate of a traitor tracing scheme is the size of a ciphertext divided by the size of the corresponding plaintext. This rate is a very important parameter, since it represents the cost of the scheme, in terms of transmission. Kiayias and Yung in [16], and later Chabanne et al. in [6], propose schemes with a constant ciphertext rate, i.e. the ciphertext rate does not depend on the number of users, or on the maximum number of traitors. In this way, their constructions are the most efficient ones. However, the ciphertexts are very long in these schemes, and a decoder cannot obtain any information about the plaintext before the end of the transmission of the ciphertext. This is a very unpleasant fact, and the ability for a decoder to decrypt progressively must be considered as an important practical feature.

Outside of their performance in terms of ciphertext rate, some schemes have additional properties. This is the case of [6], where the authors provide public traceability, i.e. the tracing procedure may be done by untrusted parties. Another new approach is proposed in [3, 5], where the authors give fully collusion resistant schemes, i.e. schemes secure against pirate decoders built by coalitions of any size. However, these last schemes have a large ciphertext rate in comparison with the previous schemes when the coalitions are small.

Except a scheme built in [14], the mentioned traitor tracing schemes are not secure against abrupt and history-recording pirate decoders.

1.2 Our contribution

Our main goal consists in building a traitor tracing scheme with a constant ciphertext rate, that is secure in a strong model of pirate decoders. First, we present a model for pirate decoders which is very close to the one given in [14], but nonetheless different. A powerful pirate decoder is history-recording, and may refuse to answer to some ciphertexts, or fake-ciphertexts used in a tracing procedure. We impose only in our definitions that this pirate decoder decrypts correctly valid ciphertexts with a minimum rate. Such pirate decoder is stronger than an abrupt and history-recording one.

Why would a pirate decoder refuse to decrypt valid ciphertexts ? The goal for such pirate decoder is to protect the traitors. The best solution is obviously to refuse to decrypt when the tracing procedure is running, and not when the pirate decoder receives normally the data. However, the pirate decoder cannot always make a distinction between valid ciphertexts and fake-ciphertexts built during the tracing procedure. The only solution for the pirate decoder is then to refuse to answer to some ciphertexts, having a specific property, even if they seem (or if they are) valid ciphertexts.

This could appear useless, but in some schemes, like [12] or [16], a pirate decoder refusing the decryption over specific ciphertexts can not be traced with the given tracing procedures, although it obtains an overwhelming part of the plaintext. This problem has been identified in [16] and the authors suggest the use of a transform to prevent this behavior. In multimedia data, such loss of a short part of the data may not be too serious, like for example the loss of a few seconds in an entire movie. It can then be an acceptable price to pay to avoid the identification of the traitors.

Once the model is defined, we build a scheme secure in this model, using the strategy given in [16]: we give a basic scheme for 2 users, and we use several instances of this basic scheme to construct a scheme for a large number of users. The building of the basic scheme follows the work developed in [14], and uses watermarking. The generalization to a large number of users requires a collusion secure code with erasure, like presented in [13, 22], but their schemes are not usable with our requirements. We build then a binary collusion secure code with erasure.

The resulting scheme has a constant ciphertext size, and is proved secure against powerful pirate decoders. Like previously given schemes with constant ciphertext rate, the ciphertexts are very long. However, these ciphertexts can be decrypted progressively.

1.3 Preliminaries

In the following definitions and proofs, we implicitly use a given parameter k: all algorithms are polynomial in time in k. The level of security depend on this parameter k, since it represents the limits of the adversaries in the security models.

2 Model for traitor tracing schemes and pirate decoders

2.1 Traitor tracing schemes

The following definition for a traitor tracing scheme is very similar to previously given definitions with black-box traceability, like in [3]. The initialization function builds the keys for the authorized users, for the supplier and for the tracer. The encryption and decryption

functions allow the supplier to deliver data to all authorized users. The tracing method is designed to interact with some captured decoder: it builds queries which are then decrypted by the decoder. These queries may be valid ciphertexts, as well as fake-ciphertexts, i.e. modified ciphertexts that users may not decrypt all in the same way, or that some users may not decrypt.

Definition 1 (Traitor tracing scheme) *A traitor tracing scheme consists in four algorithms:*
- the ***initialization function*** *is a randomized algorithm, which inputs n and builds an encryption key ek, a tracing key tk, and a set of n decryption keys: $dk_1, \ldots, dk_n$,*
- the ***encryption function*** *is a randomized algorithm E, which encrypts some plaintext m using the encryption key ek,*
- the ***decryption function*** *is an algorithm D, which decrypts some ciphertext c using a decryption key dk_u. For all user u and all message m, the compatibility requires: $D_{dk_u}(E_{ek}(m)) = m$,*
- the ***tracing method*** *is a randomized algorithm T, which builds queries for a decoder, seen as a decryption oracle, using the tracing key tk. It outputs a subset of users, from the answers given by the decoder to these queries.*

To use this traitor tracing scheme, a supplier has to launch the initialization function. The number n of users is predetermined, even if these users are not known: when a new user joins the system, the data supplier can give him a precomputed decryption key.

The previous definition can be used for a large class of traitor tracing schemes: a public-key traitor tracing scheme requires that ek is public, a public-traceable traitor tracing scheme requires that tk is public.

2.2 Pirate decoders

We introduce now pirate decoders: a pirate decoder is built by a coalition of traitors, and contains a set of decryption keys. Before its capture, a pirate decoder receives queries, which are valid ciphertexts, and it has then to decrypt these queries: the pirate decoder is in a reception world. When it is captured, however, the pirate decoder is analyzed in a specific context, where it receives queries built by the tracing procedure: the pirate is in a tracing world. In this second world, it must avoid to give any information about the decryption keys it uses. But the pirate decoder has no information about the world: the queries it receives are the only way for the pirate decoder to identify the world, and to change its behavior.

As already mentioned, Kiayias and Yung proposed in [14] strong models for pirate decoders. In their models, a pirate decoder may be history-recording, and abrupt.

A resettable pirate decoder is a pirate decoder that the tracer can reset to its initial state at will, during the tracing procedure. In this way, the tracer can be sure that the answers to its queries are independent. On the opposite, a history-recording pirate decoder can use the previous queries sent by the tracer, and its answers to these queries. Such pirate decoders are more difficult to trace, since they may try to detect the running of a tracing procedure, not only from an unique query but from the whole list of queries.

An abrupt pirate decoder may trigger some strong reactions if it detects that a tracing procedure is running. It may be the erasure of all the decryption keys contained in the pirate decoder, or the release of a virus, in order to stop the tracing procedure. In this way,

the reaction is destructive for the pirate decoder, and it must be used carefully, i.e. only in the tracing case. Moreover, Matsushita and Imai show in [18] (in a resettable setting) that a strong reaction may be detected and used by the tracer.

We consider here history-recording pirate decoders which may refuse to answer to some queries, even valid ciphertexts. In this model, a pirate decoder may, for example, decrypt queries only with a fixed probability, or refuse to decrypt queries having specific properties. As already mentioned, if the loss of a small proportion of plaintexts prevents a tracer to identify a traitor, then a pirate decoder may accept this loss, and refuse to decrypt ciphertexts that it considers as unsafe.

In this model, when such pirate decoder detects a tracing procedure, it can decide to refuse to answer to any future query it will receive: its behavior is then similar to the one of an abrupt pirate decoder. However, the tracer cannot exactly know when the pirate decoder has detected its tracing procedure: the first refusals may be arbitrary, or the pirate decoder may reply to a few more queries before stopping. Such pirate decoder is then stronger than an abrupt one.

Definition 2 (Powerful pirate decoder) *A powerful pirate decoder $\mathcal{P}$ is a randomized algorithm which is initialized with the decryption keys given to a coalition C of users $\{dk_u \,/\, u \in C\}$. On queries q_i, it computes answers a_i, possibly void (and noted $\perp$), using all past queries and answers.*

We assume in this definition that a powerful pirate decoder may access the whole set of decryption keys given to the members of the coalition. If the coalition computes some "new" decryption key and decryption mechanism, and gives them to the pirate decoder, the building of these components can be achieved by the pirate decoder in the model.

2.3 Resilience of a traitor tracing scheme

We can now define the tracing game on a pirate decoder, which will be the basis of our security definition for the traitor tracing schemes:

Definition 3 (Tracing game) *The tracing game is played between an environment Ω and a powerful pirate decoder $\mathcal{P}$ for a traitor tracing scheme:*
- Ω chooses n, and generates the keys ek, tk, $dk_1, \ldots, dk_n$, using the initialization function,
- $\mathcal{P}$ chooses a coalition C of users in $\{1, \ldots, n\}$, and gives it to Ω,
- Ω gives $\{dk_u \,/\, u \in C\}$ to $\mathcal{P}$,
- Ω uses the tracing method T with the key tk against $\mathcal{P}$: the environment Ω submits to the pirate decoder $\mathcal{P}$ the queries built by T.
The pirate decoder $\mathcal{P}$ wins the game if the subset of users output by the tracing method T is empty or not included in the coalition C.

Obviously, a pirate decoder cannot be efficiently traced if it never decrypts any query. We must make a distinction between useless and useful pirate decoders, since the tracing procedure must be efficient against useful pirate decoders, and it can clearly not be efficient against useless pirate decoders. We consider that a pirate decoder is useful if it correctly decrypts valid ciphertexts with a probability greater than a given threshold. It does not imply that this pirate decoder accepts to decrypt fake-ciphertexts with the same probability.

We can now define the resilience of a traitor tracing scheme, i.e. the capacity of the tracing procedure to identify some traitors from a powerful pirate decoder:

Definition 4 (Resilience) *A traitor tracing scheme is said α-threshold (p, t)-resilient if any pirate decoder, which decrypts valid ciphertexts with a probability greater than α, wins with a probability less than p the tracing game, when the coalition of traitors contains at most t users.*

A traitor tracing scheme is said (p, t)-resilient if it is α-threshold (p, t)-resilient for all $\alpha > 0$. For simplicity, a traitor tracing scheme is said (α-threshold) t-resilient if there exists some $p < 1$ such that it is (α-threshold) (p, t)-resilient.

2.4 Previous schemes in this model

Previous traitor tracing schemes are often not resilient with these definitions: in classic schemes based on a repartition of decryption keys between users, like in [7, 20], the black-box extraction of the decryption keys from the pirate decoder usually requires the assumption that the pirate decoder is resettable, as explained in [8]. However, when these schemes are modified like in [14], they become α-threshold (p, t)-resilient, when α is large enough.

The case of schemes based on algebraic properties, like in [2, 17], has already been considered in [15]: these schemes are not resilient if the number of traitors is superlogarithmic in the number of users, even in an easier model.

The schemes given in [12] are not resilient with these definitions, even if the fake-ciphertexts used in the tracing procedure are indistinguishable from valid ciphertexts: we can consider a pirate decoder, which uses its set of decryption keys to obtain a set of variants, and replies with the most received variant. If the variants given to the traitors are all different, then the pirate decoder refuses to answer. This pirate decoder cannot be used to obtain a traitor, with the given tracing procedures, although it correctly decrypts valid ciphertexts with a very large probability. A history-recording pirate decoder may have a more ingenious behavior: when it replies with some variant, it tries to detect if the next ciphertext corresponds to the following step of the tracing procedure. If it occurs several times, it means that its answers are used for tracing: the pirate decoder is then compromised, and may not reply anymore to queries, at least until the tracing procedure ends.

The case of constant ciphertext rate schemes, using collusion secure codes, like in [6, 16], will be precisely considered in the next section: these schemes are not resilient, even with the use of an all-or-nothing transform.

Even in [3], one of the last papers about traitor tracing, the authors explain themself that their scheme is not resilient in a strong model.

The only schemes resilient in this model are given in [14] and are not efficient. There is then a need of efficient schemes, like with constant ciphertext rate, which are resilient in this strong model.

3 Schemes based on collusion secure codes

As previously mentioned, the current schemes based on collusion secure codes are not resilient in our security model. Since we use a similar construction later, We give then some details about the construction of these schemes, mainly using [4, 16], and show precisely why these schemes are not resilient.

3.1 Collusion secure codes

Boneh and Shaw introduce collusion secure codes in [4] to fingerprint some digital data by marking differently each copy of this data. They propose to add a few bits (the marks) in the digital data: a single copy does not allow to distinguish the marks from the real data. When some users decide to collude and share their copies of the same data, they may identify the marks for which two different values have been used in the copies they own. These marks can then be modified or deleted. Boneh and Shaw present the collusion secure codes as an efficient solution for this problem: using such code for the marks, a copy built by a coalition can be used to trace back a member of this coalition.

A binary (l, n)-code is a set Γ of n elements in $\{0,1\}^l : \Gamma = \{w^{(1)}, \ldots, w^{(n)}\} \subset \{0,1\}^l$. For $u \in \{1, \ldots, n\}$, $w^{(u)}$ is the set of marks placed in the copy given to the user u. A coalition of t users is represented by a subset C of $\{1, \ldots, n\}$. For $i \in \{1, \ldots, l\}$, such coalition can detect and alterate the mark in position i only when two members of the coalition, u and u', have different values for this mark. This coalition C can then build a pirate copy with a set of marks $w \in \{0, 1, ?\}^l$ ("?" means that the mark has been altered or erased) following:

$$\forall i \in \{1, \ldots, l\}\,, \begin{cases} w_i = b \in \{0,1\} \Longrightarrow \exists u \in C \,/\, w_i^{(u)} = b\,, \\ w_i = ? \Longrightarrow \exists (u, u') \in C^2 \,/\, w_i^{(u)} \neq w_i^{(u')}\,. \end{cases}$$

The set of all possible sets of marks w verifying the previous property is denoted by $F(C)$. A (l, n)-code is said to be (p, t)-collusion secure if it can trace coalitions up to t users with probability of error at most p, i.e. there exists an efficient procedure τ, computing a set of users from a set of marks $w \in \{0, 1, ?\}^l$, such that the event $\emptyset \neq \tau(w) \subset C$ occurs with a probability larger than $(1 - p)$, when C contains at most t users, and when $w \in F(C)$.

3.2 Use for traitor tracing

To get an efficient traitor tracing scheme against t traitors in n users, Kiayias and Yung show in [16] that two building blocks are required: a (p, t)-collusion secure (l, n)-code, and l instances of a basic traitor tracing scheme against 1 traitor in 2 users.

Let $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$ be a (p, t)-collusion secure (l, n)-code. Let E, D and T be respectively the encryption function, the decryption function and the tracing method for a basic traitor tracing scheme. The initialization function of the global traitor tracing scheme consists in l calls to the initialization function of the basic scheme: for $1 \leq i \leq l$, let $ek^{(i)}$ be the encryption key, $(dk_1^{(i)}, dk_2^{(i)})$ be the decryption keys, and $tk^{(i)}$ be the tracing key generated for the instance i of the basic scheme. The plaintexts (resp. ciphertexts) of the following traitor tracing scheme are sequences of l plaintexts (resp. ciphertexts) of the basic scheme.

- **Encryption and tracing keys:** $\mathsf{ek} = (ek^{(1)}, \ldots, ek^{(l)})$, $\mathsf{tk} = (tk^{(1)}, \ldots, tk^{(l)})$.
- **Decryption keys:** for $u \in \{1, \ldots, n\}$, the user u obtains $\mathsf{dk}_u = (\mathsf{dk}_u^{(1)}, \ldots, \mathsf{dk}_u^{(l)})$, where:

$$\forall i \in \{1, \ldots, l\},\ \mathsf{dk}_u^{(i)} = \begin{cases} dk_1^{(i)} & \text{if } w_i^{(u)} = 0\,, \\ dk_2^{(i)} & \text{if } w_i^{(u)} = 1\,. \end{cases}$$

- **Encryption function:** $\mathcal{E}_{\mathsf{ek}}(m_1, \ldots, m_l) = (E_{ek^{(1)}}(m_1), \ldots, E_{ek^{(l)}}(m_l))$.

- **Decryption function:** $\mathcal{D}_{\mathsf{dk}_u}(c_1, \ldots, c_l) = (D_{\mathsf{dk}_u^{(1)}}(c_1), \ldots, D_{\mathsf{dk}_u^{(l)}}(c_l))$.

- **Tracing method:** For each instance i of the basic traitor tracing scheme, the procedure uses T with $tk^{(i)}$. It builds then $w \in \{0, 1, ?\}^l$:

$$w_i = \begin{cases} 0 & \text{if } T \text{ with } tk^{(i)} \text{ outputs the set } \{1\}, \\ 1 & \text{if } T \text{ with } tk^{(i)} \text{ outputs the set } \{2\}, \\ ? & \text{in the other cases.} \end{cases}$$

Since Γ is a collusion secure code, $\tau(w)$ gives a set of users who participated in the building of the decryption box with a good probability.

3.3 Properties and tracing

The cryptographic secrecy of the global traitor tracing scheme is based on the one of the basic traitor tracing scheme: if the encryption function of the basic traitor tracing scheme is secure against chosen plaintext attack, the one of the global scheme is secure against chosen plaintext attack (see part 4.1 for the definition of LoR-CPA security, for example).

The ciphertext rate of the global traitor tracing scheme is the one of the basic traitor tracing scheme (2 or 3 in [16], improved in almost 1 in [6]), and thus, it does not depend on the number of users, or on the number of traitors.

The tracing procedures of the collusion secure codes given in [4] are probabilistic: they may reply with innocent users, but with an arbitrarily low probability. The traitor tracing schemes based on these codes inherit therefore the same flaw. Moreover, a traced traitor may deny his guilt in specific constructions, as presented in a proof in [4]. The probability of error must then be carefully chosen.

The tracing procedure consists in using the basic tracing procedure on each instance. However, if the pirate decoder never replies to ciphertexts over specific instances, its decryption keys for these instances cannot be identified, and then, the word w cannot be completely built. To prevent this behavior, Kiayias and Yung propose in [16] the use of an all-or-nothing transform, obtained from [21]: the data sent using all the instances are mixed up and a pirate decoder can obtain the data only by recombining all the plaintexts. When a plaintext is missing, the data cannot be recovered, and then, a pirate decoder must use at least one decryption key for each instance.

The basic traitor tracing schemes used in [6, 16] are however not resilient against abrupt or powerful pirate decoder, since the fake-ciphertexts used in the tracing procedure are not valid ciphertexts. In this way, with or without the all-or-nothing transform, the global scheme cannot be resilient against abrupt or powerful pirate decoder for the same reason. Even worse, the all-or-nothing transform is not compatible with the use of a basic traitor tracing scheme resilient against abrupt or powerful pirate decoders: in such case, valid ciphertexts must lead to different plaintexts, depending on the decryption key used (see part 4.1), although the all-or-nothing transform requires the knowledge of a precise plaintext.

We consider now the scheme from [16] using an all-or-nothing transform, which is the most "secure" one, even if it is not resilient with strong definitions. The size of the ciphertext for the global traitor tracing scheme is multiplied by l, which is a very large factor, and the size of all keys as well. For example, in [7], Chor et al. consider their schemes for $t = 32$ traitors, $n = 10^9$ users with an error probability $p = 2^{-10}$. Using

the logarithmic length collusion secure code proposed in [4], the number of instances l is given by: $l = O(t^4 \log(n/p) \log(t^2 \log(n/p)/p))$. With the given parameters, we obtain: $t^4 \log(n/p) \log(t^2 \log(n/p)/p) \approx 5.10^8$.

The storage of the encryption key, or of a decryption key requires then a large memory space. The most important point is that the ciphertext is very long: even if the size of the ciphertext associated to one instance is 256 bits, the size of the full ciphertext is about 15 gigabytes, which is considerable. It is embarrassing because the all-or-nothing transform requires that a decoder decrypts a whole ciphertext before obtaining any information over the plaintext: the data cannot be obtained progressively.

4 A scheme secure against powerful pirates

To build a traitor tracing scheme secure in our model, we use the same strategy than Kiayias and Yung in [16]. We need a basic traitor tracing scheme for two users, and a code similar to collusion secure codes. The building of the basic scheme follows the work of Kiayias and Yung in [14]: we use watermarking, so that only valid ciphertext are sent to the pirate decoder in the tracing procedure. For the second part, a collusion secure code is not possible, since we saw that an all-or-nothing transform, required for the codes given by Boneh and Shaw in [4], is not compatible with our basic traitor tracing scheme. We need a collusion secure code with erasure, like defined in [13, 22].

4.1 Basic traitor tracing scheme

The tracing procedure of a traitor tracing scheme must give some information to the tracer, i.e. the use of different decryption keys on a given fake-ciphertext must lead to different answers. If the standard way of sending a message always leads to an unique plaintext, a pirate decoder which sees different possible answers to a given query knows that this query is part of the tracing procedure. To avoid this problem, the standard sending of a message must as well lead to different plaintexts.

This solution seems paradoxical, since the initial goal is to securely send the same data to multiple users. But Fiat and Tassa already explore this way in [12], using watermarking techniques, and Kiayias and Yung use the same notion in [14]. The use of watermarking implies a strong restriction: only multimedia data (like music, images, movies...) can be slightly modified in order to obtain different versions, with almost the same content. In this way, we cannot use this method for cryptographic keys, in order to directly build a hybrid scheme.

We adapt the formalism and the ideas developed in [9, 10]. The acceptance function A is a way to decide if two messages are close to each other or not. If $\mathsf{A}(m, m')$ is true, then m and m' are close, and m' is an acceptable message for m. If a distance is defined over the space of the messages, then a good definition for A would be: $\mathsf{A}(m, m') = \text{true}$ if and only if $\mathit{distance}(m, m') \leq \lambda$. The generation function outputs two variants of a given message m, such that these variants are acceptable for m. The detection function identifies the variant from which m' might come: it replies with 0 when it fails.

Definition 5 (Watermarking scheme) *A watermarking scheme (for 2 users) consists in three algorithms:*
- the ***acceptance function*** *is a deterministic algorithm* A*, which takes as input a pair of messages* (m, m') *and replies with* $\mathsf{A}(m, m') \in \{true, false\}$*,*
- the ***generation function*** *is a randomized algorithm* G*, which takes as input a message* m*, and builds two variants* m_1 *and* m_2 *such that* $\mathsf{A}(m, m_1) = \mathsf{A}(m, m_2) = true$*,*
- the ***detection function*** *is a deterministic algorithm* D*, which takes as input four messages:* m*,* m_1*,* m_2*, and* m'*, where* m_1 *and* m_2 *are two variants of* m*, and replies with an index* $u' \in \{0, 1, 2\}$*.*

An adversary of a watermarking scheme receives a variant m_u generated from a message m, and builds a message m'. Its goal is to build a message m' acceptable for m, but which is not detected as u.

Definition 6 (Adversary of a watermarking scheme) *An adversary of a watermarking scheme is a randomized algorithm which knows the watermarking scheme (*A*,* G *and* D*). It receives a message* m_u *and computes a message* m' *from it.*

We can now define the detection game and obtain from this game a definition for the robustness of the watermarking scheme:

Definition 7 (Robustness) *The detection game is played between an environment* Ω *and an adversary* $\mathcal{A}$ *for a watermarking scheme, on a given message* m*:*
- Ω *computes* $(m_1, m_2) = \mathsf{G}(m)$*, randomly chooses* $u \in \{1, 2\}$ *and gives* m_u *to* $\mathcal{A}$*,*
- $\mathcal{A}$ *replies with* m'*,*
- Ω *computes* $u' = \mathsf{D}(m', m, m_1, m_2)$*,*
The adversary $\mathcal{A}$ *wins the game if* $u' = \bar{u}$ *(false detection), or if* $u' = 0$ *with* $\mathsf{A}(m, m') = true$ *(no detection).*

A watermarking scheme is said (p_w, q_w)*-robust if for all message* m*, for all adversary* $\mathcal{A}$*:*
- the probability that $\mathcal{A}$ *wins the detection game with* $u' = \bar{u}$ *is less than* p_w*,*
- the probability that $\mathcal{A}$ *wins the detection game with* $u' = 0$ *and* $\mathsf{A}(m, m') = true$ *is less than* q_w*.*

The first part of the definition means that the detection function identifies the wrong variant with a probability at most p_w. The second one means that the detection function identifies some variant with a probability greater than $(1 - q_w)$ when the adversary replies with an acceptable message. This robust watermarking scheme can be used with an encryption scheme to build a basic traitor tracing scheme. The following notion of security comes from the definition given in [1] for the symmetric encryption:

Definition 8 (LoR-CPA Security) *The Left-or-Right (LoR) game is played between an environment* Ω *and an adversary* $\mathcal{A}$ *for an encryption function* E*:*
- Ω *randomly chooses* $v \in \{1, 2\}$ *and a pair of keys* (ek, dk)*,*
- Ω *gives* ek *to* $\mathcal{A}$ *(only in the case of a public-key encryption),*
- $\mathcal{A}$ *sends pairs of messages* (m_1, m_2) *to* Ω *which replies with* $\mathsf{E}_{ek}(m_v)$*,*
- $\mathcal{A}$ *outputs* $v' \in \{1, 2\}$*.*
The adversary $\mathcal{A}$ *wins the game if* $v' = v$*.*

An encryption scheme E *is* adv_e*-LoR-CPA secure if for all adversary* $\mathcal{A}$ *in the LoR game,* $|\Pr[\mathcal{A}\ wins] - \Pr[\mathcal{A}\ loses]| \leq adv_e$.

We define the following obvious traitor tracing scheme, using a watermarking scheme and an encryption scheme:

Definition 9 (Basic traitor tracing scheme) *Let* E *be an* adv_e*-LoR-CPA encryption scheme, and* $(\mathsf{A}, \mathsf{G}, \mathsf{D})$ *be a* (p_w, q_w)*-robust watermarking scheme. The basic traitor tracing scheme is defined by the following algorithms:*
- the initialization generates two pairs of keys, (ek_1, dk_1) *and* (ek_2, dk_2)*, for the encryption* E*. The encryption key is* $ek = (ek_1, ek_2)$*, the decryption keys are* dk_1 *and* dk_2*, and the tracing key is* $tk = ek$*,*
- to encrypt some message m*, the supplier obtains two variants* m_1 *and* m_2 *using* G*, and randomly chooses a permutation* σ *over* $\{1, 2\}$*:* $E_{ek}(m) = (\mathsf{E}_{ek_1}(m_{\sigma(1)}), \mathsf{E}_{ek_2}(m_{\sigma(2)}))$*,*
- the user u *obtains the acceptable plaintext* $m_{\sigma(u)}$ *using its decryption key* dk_u*,*
- the tracing procedure builds valid ciphertexts, until the pirate decoder has answered N_p *times with an acceptable plaintext. It computes* $u' = \sigma(\mathsf{D}(m', m, m_1, m_2))$ *on these plaintexts and replies with the most found index in* $\{1, 2\}$ *(ties are broken arbitrarily).* N_p *depends on the parameter* $p \in]2adv_e, 1[$*:*

$$N_p = \max\left(3\left\lceil \frac{\log(p/2 - adv_e)}{\log(8\max(p_w, q_w))} \right\rceil, 1\right).$$

Remark 1. The tracing procedure is valid against any pirate decoder which correctly decrypts valid ciphertexts with a probability greater than any threshold $\alpha > 0$, since about (N_p/α) queries give rise to N_p acceptable plaintexts.

Remark 2. The advantage adv_e may depend on the number of queries that an adversary can make in the Left-or-Right game. The number of queries used in the tracing procedure is at least N_p, which depends by definition on adv_e: N_p must be lower than the number of queries corresponding to adv_e. Since N_p is logarithmic in adv_e, no problem should arise from this double dependency.

Theorem 1 *Let* $(\mathsf{A}, \mathsf{G}, \mathsf{D})$ *be a* (p_w, q_w)*-robust watermarking scheme, and* E *be an* adv_e*-LoR-CPA encryption scheme. For all* $p > 2adv_e$*, for all* $\alpha > 0$*, the previous construction is a* α*-threshold* $(p, 2)$*-resilient traitor tracing scheme for* 2 *users if* $q_w < 1/8$ *and* $p_w < 1/8$.

Proof. This theorem is proved in appendix A.1.

We already explained that we cannot use such scheme for hybrid encryption, since secret keys cannot be modified. However, we can modify the previous scheme to obtain a hybrid scheme, under the assumption that the data can be watermarked. In this way, with an efficient symmetrical encryption S, the ciphertext for the data m would be: $E_{ek}(m) = (\mathsf{E}_{ek_1}(sk_1), \mathsf{E}_{ek_2}(sk_2), \mathsf{S}_{sk_1}(m_{\sigma(1)}), \mathsf{S}_{sk_2}(m_{\sigma(2)}))$, where sk_1 and sk_2 are two randomly chosen secret keys for S.

Variants of a message have almost the same size of the message. With efficient encryption schemes, the ciphertext rate of the basic traitor tracing scheme is about 2. The hybrid variant has quite the same ciphertext rate, if we consider that the size of the keys are short in comparison with the size of the messages.

4.2 Collusion secure codes with erasure

In our definitions of pirate decoder and threshold resilience, the pirate decoder may refuse to reply to some requests. It has only to reply with a rate larger than a given threshold. In the collusion secure codes context, it means that a coalition may destroy a part of the data to remove some marks: these removed marks may prevent the tracer from finding a traitor with a good probability.

Codes secure against collusion and against such erasures have already been studied. Error-tolerant collusion secure codes have been defined by Guth and Pfitzmann in [13]. However, their schemes assume that the erasures are randomly distributed among the bits of the fingerprint. Safavi-Naini and Wang propose in [22] a q-ary version of this definition, and use a similar assumption for some of their schemes. Since the adversary may take advantage from its knowledge of different words in the code to erase specific marking positions, we have to remain in the general case. The following definition is then a binary version of the definition given in [22], and a version without extra assumption of the one given in [13]:

Definition 10 (Collusion secure code with erasure - binary version) *We consider a (l, n)-code Γ, i.e. a set of n elements in $\{0,1\}^l$: $\Gamma = \{w^{(1)}, \ldots, w^{(n)}\}$. A tracing procedure for this code is an efficient procedure τ which gives a set of users in $\{1, \ldots, n\}$ from an input in $\{0, 1, \perp\}^l$.*

A t-adversary with α-erasure is a randomized algorithm $\mathcal{A}$, which chooses first a coalition $C \subset \{1, \ldots, n\}$, of size at most t. $\mathcal{A}$ obtains then $\{w^{(u)} \,/\, u \in C\}$ and computes $w \in \{0, 1, \perp\}^l$ such that:

$$w \in F_\alpha(C), i.e. \begin{cases} \forall i \in \{1, \ldots, l\}\,,\ w_i = b \in \{0,1\} \Longrightarrow \exists u \in C \,/\, w_i^{(u)} = b\,, \\ \#\{i \in \{1, \ldots, l\} \,/\, w_i \neq \perp\} \geq \alpha\, l\,. \end{cases}$$

The t-adversary with α-erasure $\mathcal{A}$ wins if it outputs w such that $\tau(w)$ is empty or not included in the chosen coalition C.

The code Γ is α-erasure (p, t)-collusion secure with the tracing procedure τ if any t-adversary with α-erasure $\mathcal{A}$ wins against (Γ, τ) with a probability less than p.

This definition is stronger than the one of a collusion secure code: for all α, a α-erasure (p, t)-collusion secure code is (p, t)-collusion secure in the sense given in [4]. To trace $w \in \{0, 1, ?\}^l$, we first compute w', replacing ? by 0 in w, it is then enough to trace w' which is a valid output for a t-adversary with α-erasure.

The codes for shortened fingerprints presented in [22, 23] have stronger requirements than what is required in our definition: in this case, the tracer obtains a shortened word, and has first to identify the positions of the erasures. However, the codes for shortened fingerprints presented in [22, 23] as well as collusion secure codes with erasure presented in [22] are q-ary codes, where q is quite large. Since the ciphertext rate of the global traitor tracing schemes would be linear in q, this is not acceptable. We build then a binary α-erasure (p, t)-collusion secure code.

The following code is the code $\Gamma_0(n, d)$ presented in [4], but with a different tracing procedure $\tau_0(\alpha, \beta, p)$. Let $l = nd - d$, let $(\Pi_j)_{1 \leq j \leq n-1}$ be a random partition of the set $\{1, \ldots, l\}$ into $(n-1)$ subsets of same size d. The code $\Gamma_0(n, d)$ is the set $\{w^{(1)}, \ldots, w^{(n)}\}$ where:

$$\forall u \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n-1\}, \forall i \in \Pi_j, w_i^{(u)} = \begin{cases} 0 & \text{if } j < u\,, \\ 1 & \text{if } j \geq u\,. \end{cases}$$

The first user is the only one to receive the value 1 for the marks in Π_1. The last user is the only one to receive the value 0 for the marks in Π_{n-1}. The user $u \in \{2, \ldots, n-1\}$ is the only one to receive different values for the marks in Π_{u-1} and for those in Π_u.

The tracing procedure $\tau_0(\alpha, \beta, p)$ begins with checking the validity of the answer: if $w \in \{0, 1, \perp\}^l$ is not a valid answer for an adversary with α-erasure, the tracing procedure stops. Else, it checks if w contains at least $\lfloor \beta\, d \rfloor$ marks in each part Π_j. If it is the case, the tracing procedure acts exactly like in [4], else, it uses the lack of marks. More precisely, the tracing procedure $\tau_0(\alpha, \beta, p)$ acts as follows on $w \in \{0, 1, \perp\}^l$:

- let $d' = \lfloor \beta\, d \rfloor$,
- $\forall j \in \{1, \ldots, n-1\}$, let $r(j) = \#\{i \in \Pi_j \,/\, w_i = \perp\}$,
- if $\sum_j r(j) > (1-\alpha)\, l$, then w is not valid, and the tracing procedure aborts,
- if for all $j \in \{1, \ldots, n-1\}$, $r(j) \leq d - d'$, then let T_j be the d' first elements i in Π_j such that $w_i \neq \perp$. We define: $c(j) = \#\{i \in T_j \,/\, w_i = 1\}$. If $c(1) > 0$, then the user 1 is pronounced as traitor. If $c(n-1) < d'$, then the user n is pronounced as traitor. The tracing procedure pronounces moreover as traitors the users $u \in \{2, \ldots, n-1\}$ such that:

$$|c(u) - c(u-1)| > 2\sqrt{(c(u) + c(u-1)) \log(n/p)},$$

- else, the tracing procedure pronounces as traitors the users $u \in \{2, \ldots, n-1\}$ such that:

$$|r(u) - r(u-1)| > 2\sqrt{(r(u) + r(u-1)) \log(n/p)}.$$

Theorem 2 *The $(nd-d, n)$-code $\Gamma_0(n, d)$ is α-erasure (p, n)-collusion secure with the tracing procedure $\tau_0(\alpha, \beta, p)$ if the following conditions hold:*

$$\begin{cases} d \geq \frac{2(n-1)(n-2)\log(n/p)+1}{\beta} \sim \frac{2n^2 \log(n/p)}{\beta}, \\ d \geq \frac{8(1-\beta)(n-1)(n-2)\log(n/p)}{(\alpha-\beta)^2} \sim \frac{8(1-\beta)n^2 \log(n/p)}{(\alpha-\beta)^2}. \end{cases}$$

Proof. This theorem is proved in appendix A.2.

Remark 3. We can use any $\beta \in]0, \alpha[$ in the tracing procedure. However, the first lower-bound for d given in theorem 2 is large when β is close to 0, and the second lower-bound is large when β is close to α. The value of β must be carefully chosen, and $\frac{\alpha+2}{5} - \frac{2\sqrt{1+\alpha-\alpha^2}}{5}$ is a good choice for β when n is large.

4.3 Logarithm length adaptation

We use the strategy given in [4, 11] to build a collusion secure code with erasure, with a logarithmic length in the number of users. We build the $((2t-1)d\rho, n)$-code $\Gamma(n, \rho, t, d)$ as follows:

- let $(\Gamma_j(2t, d))_{1 \leq j \leq \rho}$ be ρ instances of the code $\Gamma_0(2t, d)$, built using different partitions: for all j in $\{1, \ldots, \rho\}$, $\Gamma_j(2t, d) = \{w^{(j,1)}, \ldots, w^{(j,2t)}\} \subset \{0, 1\}^{(2t-1)d}$,
- let $(h_j)_{1 \leq j \leq \rho}$ be ρ randomly built functions, from $\{1, \ldots, n\}$ into $\{1, \ldots, 2t\}$: for all j in $\{1, \ldots, \rho\}$, for all u in $\{1, \ldots, n\}$, $h_j(u)$ is randomly chosen in $\{1, \ldots, 2t\}$,
- for all user u in $\{1, \ldots, n\}$, $w^{(u)}$ is the concatenation of $(w^{(j, h_j(u))})_{1 \leq j \leq \rho}$.

The tracing procedure $\tau(\alpha, \beta, \gamma, p)$ cuts the answer in ρ parts, traces these parts with the tracing procedures $\tau_j(\alpha, \beta, p/(2\rho))$ associated to the codes $(\Gamma_j(2t, d))$, and keeps at most one output for each part: x_j (if there is no output, $x_j = 0$). It replies with the user $u \in \{1, \ldots, n\}$ such that $h_j(u) = x_j$ in the most number of parts among the first $(\gamma - \alpha)\rho/(1 - \alpha)$ parts where $x_j \neq 0$ (ties are broken arbitrarily).

Theorem 3 *If the code $\Gamma_0(2t, d)$, together with the tracing procedure $\tau_0(\alpha, \beta, p/(2\rho))$, is α-erasure $(p/(2\rho), 2t)$-collusion secure, then for all $\gamma > \alpha$, the $((2t-1)d\rho, n)$-code $\Gamma(n, \rho, t, d)$ is γ-erasure (p, t)-collusion secure with the tracing procedure $\tau(\alpha, \beta, \gamma, p)$ if:*

$$\rho \geq \frac{4(1-\alpha)t \, \log(4n/p)}{3(\gamma - \alpha)\log(3/2)}.$$

Proof. This theorem is proved in appendix A.3.

The length of the γ-erasure (p, t)-collusion secure code $\Gamma(n, \rho, t, d)$ grows then asymptotically like $t^4 \log(n/p) \log(t^2 \log(n/p)/p)$. This code is then very pleasant for a large number of users, with a relatively small number of traitors.

4.4 The *t*-resilient-traitor tracing scheme

We can now combine the basic traitor tracing scheme and the collusion secure code with erasure to obtain a threshold traitor tracing scheme, like presented in part 3.2.

Theorem 4 *We can build a α-threshold (p, t)-resilient traitor tracing scheme for n users from:*
- a α_b-threshold $(p_b, 2)$-resilient traitor tracing scheme for 2 users,
- a α_c-erasure (p_c, t)-collusion secure (l, n)-code,
where: $\alpha = 1 - (1 - \alpha_b)(1 - \alpha_c)$ and $p = p_c + l\, p_b$.

Proof. This theorem is proved in appendix A.4.

The scheme for n users has the same ciphertext rate than the basic one (almost 2). The size of the encryption key, of the tracing key, of the decryption keys, as well as the size of the plaintexts and the ciphertexts are multiplied by l. However, the instances of the basic traitor tracing scheme are used independently, and the data can then be obtained progressively.

5 Conclusion

In this paper, we built a new traitor tracing scheme, secure in a strong security model, which has moreover a constant ciphertext rate. The size of the ciphertext is quite long, but a ciphertext can be decrypted progressively. This scheme can be improved with the use of a better collusion secure code with erasure: the composition theorem (theorem 4) gives the way to directly obtain a traitor tracing scheme from the code.

References

1. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proc. of Foundations of Computer Science (FOCS'97)*, page 394, 1997.
2. Dan Boneh and Matthew Franklin. An efficient public key traitor tracing scheme. In *Proc. of Advances in Cryptology – Crypto'99*, pages 338–353, 1999.
3. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Proc. of Advances in Cryptology – Eurocrypt'06*, pages 573–592, 2006.
4. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. In *Proc. of Advances in Cryptology – Crypto'95*, pages 452–465, 1995.
5. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proc. of ACM-CCS'06*, pages ?–?, 2006.
6. Herve Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *Proc. of Advances in Cryptology – Eurocrypt'05*, pages 542–558, 2005.
7. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Proc. of Advances in Cryptology – Crypto'94*, pages 257–270, 1994.
8. Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46:893–910, 2000.
9. Ingemar J. Cox, Joe Killian, Tom Leighton, and Talal Shamoon. A secure, robust watermark for multimedia. In *Proc. of Information Hiding (IH'96)*, pages 257–270, 1996.
10. Funda Ergun, Joe Kilian, and Ravi Kumar. A note on the limits of collusion-resistant watermarks. In *Proc. of Advances in Cryptology – Eurocrypt'99*, pages 140–149, 1999.
11. Amos Fiat and Moni Naor. Broadcast encryption. In *Proc. of Advances in Cryptology – Crypto'93*, pages 480–491, 1993.
12. Amos Fiat and Tamir Tassa. Dynamic traitor tracing. In *Proc. of Advances in Cryptology – Crypto'99*, pages 354–371, 1999.
13. Hans-Jurgen Guth and Birgit Pfitzmann. Error- and collusion-secure fingerprinting for digital data. In *Proc. of Information Hiding (IH'99)*, pages 134–145, 1999.
14. Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In *Proc. of Security and Privacy in Digital Right Management (DRM'01)*, pages 22–39, 2001.
15. Aggelos Kiayias and Moti Yung. Self protecting pirates and black-box traitor tracing. In *Proc. of Advances in Cryptology – Crypto'01*, pages 63–79, 2001.
16. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *Proc. of Advances in Cryptology – Eurocrypt'02*, pages 450–465, 2002.
17. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *Proc. of 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 *Proc. of Advances in Cryptology – Asiacrypt'04*, pages 260–275, 2004.
19. Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans.*, E82–??(1), January 1999.
20. Moni Naor and Benny Pinkas. Threshold traitor tracing. In *Proc. of Advances in Cryptology – Crypto'98*, pages 502–517, 1998.
21. Ronald Rivest. All-or-nothing encryption and the package transform. In *Proc. of Fast Software Encryption'97*, pages 210–218, 1997.
22. Reihaneh Safavi-Naini and Yejing Wang. Collusion secure q-ary fingerprinting for perceptual content. In *Proc. of Security and Privacy in Digital Right Management (DRM'01)*, pages 57–75, 2001.
23. Reihaneh Safavi-Naini and Yejing Wang. Traitor tracing for shortened and corrupted fingerprints. In *Proc. of Security and Privacy in Digital Right Management (DRM'02)*, pages 81–100, 2002.
24. V. D. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *Proc. of Security and Privacy in Digital Right Management (DRM'03)*, pages 67–76, 2003.

A Proofs

A.1 Proof of theorem 1

Theorem 1 *Let* $(\mathsf{A}, \mathsf{G}, \mathsf{D})$ *be a* (p_w, q_w)*-robust watermarking scheme, and* E *be an* adv_e*-LoR-CPA encryption scheme. For all* $p > 2adv_e$*, for all* $\alpha > 0$*, the construction presented*

in part 4.1 is a α-threshold $(p,2)$-resilient traitor tracing scheme for 2 users if $q_w < 1/8$ and $p_w < 1/8$.

If the coalition contains both users, any non-empty answer of the tracing procedure is valid. Since the tracing procedure always answer with an unique user, the tracing procedure never fails. We now consider the case of a coalition containing a single user.

Let $\mathcal{G}_0$ be the tracing game for the previously defined basic traitor tracing scheme:

- Ω generates two pairs of keys (ek_1, dk_1) and (ek_2, dk_2) for the encryption scheme E,
- $\mathcal{P}$ chooses $u \in \{1,2\}$, sends u to Ω and receives dk_u from Ω,
- Ω chooses a message m and computes $(m_1, m_2) = \mathsf{G}(m)$. Ω randomly chooses a permutation σ over $\{1,2\}$, and sends $(E_{ek_u}(m_{\sigma(u)}), E_{ek_{\bar{u}}}(m_{\sigma(\bar{u})}))$ to $\mathcal{P}$. The pirate decoder $\mathcal{P}$ replies with m': if $\mathsf{A}(m,m')$ is true, Ω computes $u' = \sigma(\mathsf{D}(m', m, m_1, m_2))$. Ω continues with this kind of queries until it has computed N_p times u'.

The pirate decoder $\mathcal{P}$ wins the game if the most found value for u' in $\{1,2\}$ is $\bar{u}$.

Let $\mathcal{G}_1$ be the same game as $\mathcal{G}_0$, but where Ω sends $(E_{ek_u}(m_{\sigma(u)}), E_{ek_{\bar{u}}}(m_{\sigma(u)}))$ instead of $(E_{ek_u}(m_u), E_{ek_{\bar{u}}}(m_{\sigma(\bar{u})}))$.

We build an adversary $\mathcal{A}$ of the LoR-CPA security of E from a pirate decoder $\mathcal{P}$:

- $\mathcal{A}$ receives ek from the environment, randomly chooses $b \in \{1,2\}$, computes a pair of keys (ek_b, dk_b) for E, and defines $ek_{\bar{b}} = ek$,
- $\mathcal{P}$ chooses $u \in \{1,2\}$ and gives u to $\mathcal{A}$: if $u \neq b$, then $\mathcal{A}$ replies to the environment with a random element in $\{1,2\}$ and stops. Otherwise, $\mathcal{A}$ gives dk_u to $\mathcal{P}$ and continues the game,
- $\mathcal{A}$ chooses a message m, computes $(m_1, m_2) = \mathsf{G}(m)$. $\mathcal{A}$ randomly chooses a permutation σ over $\{1,2\}$, and submits $(m_{\sigma(1)}, m_{\sigma(2)})$ to the environment. The environment replies with $\mathsf{E}_{ek}(m_{\sigma(v)})$, where v is fixed. $\mathcal{A}$ sends $(E_{ek_u}(m_{\sigma(u)}), E_{ek}(m_{\sigma(v)}))$ to $\mathcal{P}$, which replies with m': if $\mathsf{A}(m,m')$ is true, $\mathcal{A}$ computes $u' = \sigma(\mathsf{D}(m', m, m_1, m_2))$. $\mathcal{A}$ continues with this kind of queries until it has computed N_p times u',
- $\mathcal{A}$ replies to the environment with v', the most found value for u' in $\{1,2\}$.

Since E is an adv_e-LoR-CPA encryption scheme,

$$\begin{aligned} adv_e &\geq |\ \Pr[v' = \bar{u}\ /\ (v = \bar{u} \text{ and } u = b)] - \Pr[v' = \bar{u}\ /\ (v = u \text{ and } u = b)]\ |\ /\ 2\,, \\ &\geq |\ \Pr[v' = \bar{u} \text{ in game } \mathcal{G}_0] - \Pr[v' = \bar{u} \text{ in game } \mathcal{G}_1]\ |\ /\ 2\,, \\ &\geq |\ \Pr[\mathcal{P} \text{ wins the game } \mathcal{G}_0] - \Pr[\mathcal{P} \text{ wins the game } \mathcal{G}_1]\ |\ /\ 2\,. \end{aligned}$$

Let $\mathcal{G}_w$ be the following game between an environment Ω_w, and an adversary $\mathcal{A}_w$ of the watermarking scheme $(\mathsf{A}, \mathsf{G}, \mathsf{D})$:

- The adversary $\mathcal{A}_w$ chooses $u \in \{1,2\}$,
- Ω_w chooses a message m, computes $(m_1, m_2) = \mathsf{G}(m)$. Ω_w randomly chooses a permutation σ over $\{1,2\}$, and sends $m_{\sigma(u)}$ to $\mathcal{A}_w$. The adversary $\mathcal{A}_w$ replies with m': if $\mathsf{A}(m,m')$ is true, Ω_w computes $u' = \sigma(\mathsf{D}(m', m, m_1, m_2))$. Ω_w continues with this kind of queries until it has computed N_p times u'.

The adversary $\mathcal{A}_w$ wins the game if the most found value for u' in $\{1,2\}$ is $\bar{u}$.

The adversary $\mathcal{A}_w$ may simulate past queries: it can thus obtain no information from the past queries, and then, the game $\mathcal{G}_w$ is N_p independent instances of the detection game for the watermarking scheme $(\mathsf{A}, \mathsf{G}, \mathsf{D})$. In the game $\mathcal{G}_w$, let $N_p(0)$ be the number of answers given by $\mathcal{A}_w$ such that $u' = 0$, and $N_p(\bar{u})$ be the number of answers given by $\mathcal{A}_w$ such that $u' = \bar{u}$:

$$\begin{aligned}
\Pr[\mathcal{A}_w \text{ wins the game } \mathcal{G}_w] &\leq \Pr[N_p(0) \geq N_p/3] + \Pr[N_p(\bar{u}) \geq N_p/3]\,, \\
&\leq \sum_{i=N_p/3}^{N_p} \binom{N_p}{i} {q_w}^i + \sum_{i=N_p/3}^{N_p} \binom{N_p}{i} {p_w}^i\,, \\
&\leq 2^{N_p} {q_w}^{N_p/3} + 2^{N_p} {p_w}^{N_p/3}\,, \\
&\leq (8\,q_w)^{N_p/3} + (8\,p_w)^{N_p/3}\,, \\
&\leq p - 2adv_e\,.
\end{aligned}$$

From any pirate decoder in the game $\mathcal{G}_1$, we can build an adversary in the game $\mathcal{G}_w$ with the same probability of win, which chooses pairs of keys, uses them to encrypt the received messages and gives them to the pirate decoder in $\mathcal{G}_1$. From any adversary in $\mathcal{G}_w$, we can build a pirate decoder in $\mathcal{G}_1$ with the same probability of win, which uses its private key to decrypt the received messages and gives them to the adversary in $\mathcal{G}_w$. Then,

$$\Pr[\mathcal{P} \text{ wins the game } \mathcal{G}_0] \leq 2adv_e + \Pr[\mathcal{P} \text{ wins the game } \mathcal{G}_1] \leq 2adv_e + (p - 2adv_e) = p. \quad \blacksquare$$

A.2 Proof of theorem 2

Theorem 2 *The $(nd-d, n)$-code $\Gamma_0(n, d)$ is α-erasure (p, n)-collusion secure with the tracing procedure $\tau_0(\alpha, \beta, p)$ if the following conditions hold:*

$$\begin{cases} d \geq \frac{2(n-1)(n-2)\log(n/p)+1}{\beta} \sim \frac{2n^2\log(n/p)}{\beta}\,, \\ d \geq \frac{8(1-\beta)(n-1)(n-2)\log(n/p)}{(\alpha-\beta)^2} \sim \frac{8(1-\beta)n^2\log(n/p)}{(\alpha-\beta)^2}\,. \end{cases}$$

Let $w \in \{0, 1, \perp\}^l$ be the output given by a α-threshold n-adversary. By definition of a α-threshold n-adversary, w verifies: $\sum_j r(j) \leq (1-\alpha)\,l$.

First case : $\forall j \in \{1, \ldots, n-1\}, r(j) \leq d - d'$

For all user $u \in \{2, \ldots, n-1\}$ not in the coalition chosen by the α-threshold n-adversary, the marks in Π_{u-1} and those in Π_u are indistinguishable for the adversary, since all users in the coalition received the same values for these marks. Thus,

$$\forall k \in \{0, \ldots, 2d'\},\ \forall z \in \{0, \ldots, k\},\ \Pr[c(u) = z \,/\, c(u) + c(u-1) = k] = \frac{\binom{d'}{z}\binom{d'}{k-z}}{\binom{2d'}{k}}.$$

We denote the previous probability by $f(k, z)$. When $z \geq k/2$,

$$f(k, z+1) = \frac{\binom{d'}{z+1}}{\binom{d'}{z}} \frac{\binom{d'}{k-z-1}}{\binom{d'}{k-z}}\, f(k, z) = \frac{(d'-z)(k-z)}{(z+1)(d'-k+z+1)}\, f(k, z) \leq \frac{k-z}{z+1}\, f(k, z).$$

For all $y \geq 0$, let $I_y(k) = \Pr[|c(u) - k/2| > y \;/\; c(u) + c(u-1) = k]$. By definition, $I_y(k)$ is twice the sum of the $f(k,z)$ where $z > y + k/2$. Then,

$$I_{y+1}(k) \;=\; 2 \sum_{z > y+k/2} \frac{(d'-z)(k-z)}{(z+1)(d'-k+z+1)}\, f(k,z) \;\leq\; \frac{k/2-y}{k/2+y+1}\, I_y(k).$$

Since $\forall x \in [0,1], (1-x)/(1+x) \leq e^{-2x}$, we have: $\dfrac{k/2-y}{k/2+y+1} \;\leq\; e^{-2\frac{2y+1}{k+1}}$.

We can deduce the following recursive inequality: $I_{y+1}(k) \leq I_y(k)\, e^{\frac{-2(2y+1)}{k+1}}$.

We obtain then: $\forall b \in [0,1[, \forall i \in \mathbb{N},\, I_{b+i}(k) \leq I_b(k)\, e^{\frac{-2i(2b+i)}{k+1}} \leq e^{\frac{-2i(2b+i)}{k+1}}$.

When $y = b + i \geq \sqrt{(k+1)\log(n/p)/2 + 1}$, the previous inequality gives: $I_y(k) \leq p/n$.

With $n \geq pe^2$, $I_0(0) = 0$, $I_{\sqrt{\log(n/p)}}(1) = 0$, and $\sqrt{(k+1)\log(n/p)/2+1} \geq \sqrt{k\log(n/p)}$.

When $y \geq \sqrt{k\log(n/p)}$, we have then: $I_y(k) \leq p/n$. From the definition of $I_y(k)$:

$$\Pr\left[|c(u) - c(u-1)| > 2\sqrt{(c(u)+c(u-1))\log(n/p)}\right] \leq p/n.$$

The probability to pronounce as traitor some user not in the coalition is then upper-bounded by p (the case of users 1 and n is clear). We now prove that the tracing procedure always detects some user.

Assumption: the tracing procedure outputs no traitor.

The assumption means: $\forall j \in \{2,\ldots,n-1\}, |c(j) - c(j-1)| \leq 2\sqrt{\log(n/p)(c(j)+c(j-1))}$, $c(1) = 0$, and $c(n-1) = d'$. Let $g(j) = 1/4 + c(j)/(2\log(n/p))$. We obtain the inequality: $(g(j) - g(j-1))^2 \leq 2g(j) + 2g(j-1) - 1$.

We deduce: $(g(j) - g(j-1) - 1)^2 \leq 4g(j-1)$, and finally: $\sqrt{g(j)} \leq \sqrt{g(j-1)} + 1$. We have now an inequality between $g(1)$ and $g(n-1)$: $g(n-1) \leq (\sqrt{g(1)} + n - 2)^2$. The values of $c(1)$ and $c(n-1)$ give: $d' \leq 2(n-1)(n-2)\log(n/p)$.

The assumption is then false if $d' > 2(n-1)(n-2)\log(n/p)$. We can then conclude that the procedure always detects some traitor if:

$$d \geq \frac{2(n-1)(n-2)\log(n/p) + 1}{\beta} \sim \frac{2n^2\log(n/p)}{\beta}.$$

Second case : $r(j_{max}) > d - d'$ for some $j_{max} \in \{1,\ldots,n-1\}$

Since $\sum_j r(j) \leq (1-\alpha)\, d$, we have $r(j_{min}) \leq (1-\alpha)\, d$ for some $j_{min} \in \{1,\ldots,n-1\}$. In this proof, we assume without loss of generality that $j_{max} > j_{min}$.

We use the same strategy than in the previous case: for all user $u \in \{2,\ldots,n-1\}$ not in the coalition, the marks in Π_{u-1} and those in Π_u are indistinguishable for the adversary, and thus, when $n \geq pe^2$,

$$\Pr\left[|r(u) - r(u-1)| > 2\sqrt{(r(u)+r(u-1))\log(n/p)}\right] \leq p/n.$$

The probability to pronounce as traitor some user not in the coalition is upper-bounded by p. We now prove that the tracing procedure always detects some user.

Assumption: the tracing procedure outputs no traitor.
The same computation as in the first case gives: $\sqrt{g(j_{max})} \leq \sqrt{g(j_{min})} + n - 2$, where $g(j) = 1/4 + r(j)/(2\log(n/p))$. Since $r(j_{max}) > (1-\beta)\,d$, and $r(j_{min}) \leq (1-\alpha)\,d$, we obtain:

$$\sqrt{\frac{1}{4} + \frac{(1-\beta)\,d}{2\log(n/p)}} < \sqrt{\frac{1}{4} + \frac{(1-\alpha)\,d}{2\log(n/p)}} + n - 2.$$

We deduce:

$$\frac{(\alpha-\beta)\,d}{2\log(n/p)} < (n-2)\left(\sqrt{\frac{1}{4} + \frac{(1-\beta)\,d}{2\log(n/p)}} + \sqrt{\frac{1}{4} + \frac{(1-\alpha)\,d}{2\log(n/p)}}\right) < 2(n-2)\sqrt{\frac{1}{4} + \frac{(1-\beta)\,d}{2\log(n/p)}}.$$

We obtain then:

$$\left(\frac{(\alpha-\beta)\,d}{2\log(n/p)}\right)^2 < (n-2)^2 + \frac{2(1-\beta)(n-2)^2\,d}{\log(n/p)}.$$

$$\left(\frac{(\alpha-\beta)\,d}{2\log(n/p)} - \frac{2(1-\beta)(n-2)^2}{(\alpha-\beta)}\right)^2 < (n-2)^2 + \left(\frac{2(1-\beta)(n-2)^2}{(\alpha-\beta)}\right)^2.$$

$$\frac{(\alpha-\beta)\,d}{2\log(n/p)} < (n-2)\left(\frac{2(1-\beta)(n-2)}{(\alpha-\beta)} + \sqrt{1 + \left(\frac{2(1-\beta)(n-2)}{(\alpha-\beta)}\right)^2}\right).$$

$$d < \frac{2(n-2)\log(n/p)}{(\alpha-\beta)}\left(\frac{4(1-\beta)(n-2)}{(\alpha-\beta)} + 1\right).$$

$$d < \frac{8(1-\beta)(n-1)(n-2)\log(n/p)}{(\alpha-\beta)^2}.$$

We can then conclude that the procedure always detects some traitor if:

$$d \geq \frac{8(1-\beta)(n-1)(n-2)\log(n/p)}{(\alpha-\beta)^2} \sim \frac{8(1-\beta)n^2\log(n/p)}{(\alpha-\beta)^2}. \quad \blacksquare$$

A.3 Proof of theorem 3

Theorem 3 *If the code $\Gamma_0(2t,d)$, together with the tracing procedure $\tau_0(\alpha,\beta,p/(2\rho))$, is α-erasure $(p/(2\rho),2t)$-collusion secure, then for all $\gamma > \alpha$, the $((2t-1)d\rho,n)$-code $\Gamma(n,\rho,t,d)$ is γ-erasure (p,t)-collusion secure with the tracing procedure $\tau(\alpha,\beta,\gamma,p)$ if:*

$$\rho \geq \frac{4(1-\alpha)t\,\log(4n/p)}{3(\gamma-\alpha)\log(3/2)}.$$

Let $(w(j))_{1\leq j\leq\rho}$ be the parts obtained from $w \in \{0,1,\perp\}^l$, the output given by a γ-threshold n-adversary. For all j in $\{1,\ldots,\rho\}$, we define:

$$a(j) = \frac{\#\{i \in \{1,\ldots,(2t-1)d\} \,/\, w(j)_i \neq \perp\}}{(2t-1)d}.$$

The sum of the $a(j)$ is greater than $\gamma\,\rho$, and thus,

$$\#\{j \in \{1,\ldots,\rho\} \,/\, a(j) \geq \alpha\} + \alpha\,(\rho - \#\{j \in \{1,\ldots,\rho\} \,/\, a(j) \geq \alpha\}) > \gamma\,\rho.$$

We deduce: $\#\{j \in \{1,\ldots,\rho\} \,/\, a(j) \geq \alpha\} \geq \left\lceil \frac{(\gamma-\alpha)\,\rho}{1-\alpha} \right\rceil = \rho'$.

There are at least ρ' parts $w(j)$ which are valid outputs for a γ-threshold $(2t)$-adversary against $\Gamma_j(2t,d)$. The probability that some of the valid parts are not well-traced by the tracing procedures in $\Gamma_j(2t,d)$ is less than $p/2$. We assume now that all valid parts are well-traced: there is some traitor v in $\{1,\ldots,n\}$ such that $x_j = h_j(v)$ for at least $\lceil \rho'/t \rceil$ parts among the first ρ' parts where $x_j \neq 0$.

Let u be an innocent user. For all j, since $h_j(u)$ has been randomly chosen, $h_j(u) = x_j$ occurs with a probability $1/(2t)$ when $x_j \neq 0$. Let $d(u)$ be the number of parts where $h_j(u) = x_j$ among the first ρ' parts such that $x_j \neq 0$. For all c,

$$\Pr[d(u) = c+1] = \frac{\rho' - c}{(2t-1)(c+1)} \Pr[d(u) = c].$$

When $c \geq \lfloor 3\rho'/(4t) \rfloor$, $\Pr[d(u) = c+1] \leq \frac{(4t-3)\rho' + 4t}{3(2t-1)\rho'} \Pr[d(u) = c] \leq \frac{2}{3} \Pr[d(u) = c]$.
We obtain: $\Pr[d(u) = \lceil \rho'/t \rceil] \leq (2/3)^{3\rho'/(4t)} \Pr[d(u) = \lfloor 3\rho'/(4t) \rfloor] \leq p/(4n)$.

When $c \geq \rho'/t$, $\Pr[d(u) = c+1] \leq \frac{t-1}{2t-1} \Pr[d(u) = c] \leq \frac{1}{2} \Pr[d(u) = c]$.
And then: $\Pr[d(u) \geq \lceil \rho'/t \rceil] \leq 2 \Pr[d(u) = \lceil \rho'/t \rceil] \leq p/(2n)$.

Under the assumption that all valid parts are well-traced, the user u is the output of the tracing procedure with a probability less than $p/(2n)$, i.e. the tracing procedure detects some innocent user with a probability less than $p/2$.
Globally, the tracing procedure detects some innocent user with a probability less than p. ∎

A.4 Proof of theorem 4

Theorem 4 *We can build a α-threshold (p,t)-resilient traitor tracing scheme for n users from:*
- a α_b-threshold $(p_b, 2)$-resilient traitor tracing scheme for 2 users,
- a α_c-erasure (p_c, t)-collusion secure (l, n)-code,
where: $\alpha = 1 - (1-\alpha_b)(1-\alpha_c)$ and $p = p_c + l\,p_b$.

We use the combination presented in part 3.2. Let $\mathcal{P}$ be a pirate decoder which decrypts valid ciphertexts with a probability greater than α. For each instance $i \in \{1,\ldots,l\}$ of the basic traitor tracing scheme, we call $\alpha(i)$ the probability that $\mathcal{P}$ decrypts valid ciphertexts on this instance i. We have $\sum_{i=1}^{l} \alpha(i) \geq \alpha\, l$, and thus:
$\#\{i \in \{1,\ldots,l\} \,/\, \alpha(i) \geq \alpha_b\} + \alpha_b\,(l - \#\{i \in \{1,\ldots,l\} \,/\, \alpha(i) \geq \alpha_b\}) > \alpha\, l$.

We obtain then: $\#\{i \in \{1,\ldots,l\} \,/\, \alpha(i) \geq \alpha_b\} > \frac{\alpha - \alpha_b}{1-\alpha_b}\, l = \alpha_c\, l$.

In the tracing game, the pirate decoder receives the decryption keys associated to the sets of marks $w^{(u)}$ corresponding to the members u of the coalition C. The environment builds $w \in \{0,1,\perp\}^l$ from the answers given by $\mathcal{P}$, and then traces w with the tracing

procedure for the collusion secure code. We denote by $F_\alpha(C)$ the set of possible outputs for an α-threshold adversary of the code, using $\{w^{(u)} \,/\, u \in C\}$.

If $w \notin F_\alpha(C)$, then the pirate decoder may be used to build a pirate decoder $\mathcal{P}_b$ of the basic traitor tracing scheme: $\mathcal{P}_b$ receives an encryption key from the environment, computes a code, randomly chooses an instance $i \in \{1, \ldots, n\}$, randomly computes keys for all other instances, and gives all encryption keys to $\mathcal{P}$. When $\mathcal{P}$ asks for the decryption given to a coalition, $\mathcal{P}_b$ asks for the required decryption keys for the instance i, and replies to $\mathcal{P}$. When it receives a message, $\mathcal{P}_b$ computes messages for the other instances, sends the global message to $\mathcal{P}$, and replies with the answer given by $\mathcal{P}$ in instance i. When $w \notin F_\alpha(C)$, at least one of the instances has not been well-traced, and $\mathcal{P}_b$ wins then the game with a probability greater than the one of $\mathcal{P}$ divided by l.

If $w \in F_\alpha(C)$, then the pirate decoder may be used to build an adversary of the code $\mathcal{A}_c$: $\mathcal{A}_c$ receives the set of marks for a coalition C, builds keys, and gives the relevant keys to $\mathcal{P}$. $\mathcal{A}_c$ uses the tracing procedure of the basic scheme on each instance, and replies with the set of marks $w \in F_\alpha(C)$ deduced from the tracing procedure. $\mathcal{A}_c$ wins its tracing game with the same probability than $\mathcal{P}$.

Thus, $\Pr[\mathcal{P} \text{ wins}] \leq \Pr[\mathcal{P} \text{ wins} \,/\, w \in F_\alpha(C)] + \Pr[\mathcal{P} \text{ wins} \,/\, w \notin F_\alpha(C)] \leq p_c + l\, p_b$. ∎