Traitors Collaborating in Public: Pirates 2.0

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. This work introduces a new concept of attack against traitor tracing schemes. We call attacks of this type Pirates 2.0 attacks as they result from traitors collaborating together *in a public way*. In other words, traitors do not secretly collude but display part of their secret keys in a public place; pirate decoders are then built from this public information. The distinguishing property of Pirates 2.0 attacks is that traitors only contribute partial information about their secret key material which suffices to produce (possibly imperfect) pirate decoders while allowing them to remain anonymous. The side-effect is that traitors can publish their contributed information without the risk of being traced; giving such strong incentives to some of the legitimate users to become traitors allows coalitions to attain very large sizes that were deemed unrealistic in some previously considered models of coalitions.
This paper proposes a generic model for this new threat, that we use to assess the security of some of the most famous traitor tracing schemes. We exhibit several Pirates 2.0 attacks against these schemes, providing new theoretical insights with respect to their security. We also describe practical attacks against various instances of these schemes. Eventually, we discuss possible variations on the Pirates 2.0 theme.

1 Introduction

Traitor tracing is a cryptographic primitive introduced by Chor, Fiat, and Naor in [9] in the context of secure content distribution. This context covers for instance multimedia content rental, or broadcasting to a very large number of subscribers like in pay-TV systems, mass distribution of high value DVDs, or in web-based distribution of various multimedia contents. In all of these settings, the content is encrypted before its distribution in order to prevent illegal access which helps ensuring the revenues of the distributor. To decrypt the content, every legitimate user is provided with a decryption means, commonly called decoder. The main issue faced by the distributor is the construction and dissemination of unauthorized decoders, possibly creating a parallel market.

Hardware tamper resistant solutions are often too expensive compared to the price of the offered services. Furthermore, it would not prevent an organization from breaking into one box and extracting the necessary information to build and resell unauthorized decoders.

This is where traitor tracing schemes step into the game: the key material embedded in the decoders is diversified on a user basis. Thus, decoders are 'marked' with the identity of the user and traitor tracing allows the authority to trace a user that produced a pirate decoder. Such users, called traitors, are more powerful when they collude to create a pirate decoder. In this case, traitor tracing should allow the tracing of at least one of the traitors that took part in the coalition. A trivial solution to the problem of traitor tracing is to provide every user with a randomly chosen key that identifies him and encrypt the content as many times as there are users in the system. Obviously, such a solution is totally impracticable due to bandwidth restrictions. Hence, bandwidth preservation in traitor tracing schemes is of crucial importance.

Since the seminal work of Chor, Fiat, and Naor, there have been several proposals and improvements in traitor tracing schemes. We give a few landmarks of the work in traitor tracing but this list is of course not exhaustive. Boneh and Franklin exposed an elegant algebraic construction coming with a deterministic tracing procedure in [4]. Fiat and Tassa proposed a way to dynamically remove traitors from the system once they are caught, see [12]. Kiayias and Yung gave a powerful method to turn black-box tracing against stateless decoders into black-box tracing against stateful decoders in [16]. In [6], Boneh, Sahai, and Waters introduced a full collusion traitor tracing scheme with sub-linear ciphertext size and constant size private keys. Traitor tracing schemes based on codes have been much investigated since the seminal work of Boneh and Shaw [7]: Kiayias and Yung [17] proposed a scheme with constant rate, [8, 19] relaxed the assumption that the tracer is a trusted third party, and [3, 5] recently achieved constant size ciphertexts. Among the most famous traitor tracing schemes are schemes from the NNL framework [18] as they were used as a basis to design the widely spread content protection system for HD-DVDs and Blu-ray disks called AACS [1]. These are not exactly traitor tracing schemes, but rather very efficient broadcast encryption schemes with some black-box tracing abilities.

Pirates 2.0 attacks are primarily targeted to code based schemes and schemes from the NNL framework, but might be used against other combinatoric schemes.

1.1 Collaborative Traitors: Pirates 2.0

From the point of view of the attack model for traitor tracing schemes, there has been no radical change since the introduction of the concept in [9]. One remarkable exception is Pirate Evolution from [15] which exposes a new threat against trace and revoke schemes such as [18]. In this paper, we introduce another new threat that we call Pirates 2.0 against both traitor tracing schemes and the trace and revoke schemes from [18].

The main characteristics of our new Pirates 2.0 threat are as follows:

Anonymity Guarantee: Traitors that participate in a Pirates 2.0 attack are provided with a guarantee (through the exhibition of a mathematical proof) that *they cannot be traced by the authority.*

Partial Contributions: Traitors never need to reveal their whole secret key.

Public Collusions: Traitors operate in a public environment: they publish secret data from their decoders.
Large Coalitions: Traitors take part in unusually large coalitions.
Dynamic Coalitions: Traitors can come into action only when necessary.

The anonymity guarantee together with considerations on imperfect decoders makes the basis of our attack scenario and everything else heavily relies on it. The anonymity guarantee indeed gives strong incentives to potential traitors to actually take the plunge: With ubiquitous access to the Internet, leaking secret data, say, in a peer-to-peer network without further action can be done very quickly and in a straightforward way. This makes it an appealing scenario among the ever growing [11, 10, 24] set of users hostile to the currently deployed Digital Rights Management systems (DRM). The characteristic that large coalitions can easily be achieved is therefore a direct consequence of the fact that traitors are *guaranteed not to be traced* by an authority.

Considerations on imperfect decoders are the other determinant ingredient: A pirate decoder is considered to be useful if it can decrypt (resp. decrypt with a high probability) valid ciphertexts; such a pirate decoder is called perfect (resp. imperfect) decoder. In previous work, it is assumed that a pirate decoder always decrypts ciphertexts from the tracer when it is not able to detect the presence of the tracing procedure, i.e. it is assumed that the pirate decoder is either perfect or only slightly imperfect. This assumption makes sense in the classical model of coalitions since any coalition, knowing at least one legitimate key, is able to decrypt all valid ciphertexts anyway. However, in the Pirates 2.0 setting, we show that another trade-off is possible for the pirates when the scheme uses variable length ciphertexts: the pirate decoder is only required to decrypt ciphertexts reasonably shaped. As an example of this scenario, consider the NNL scheme where the expansion of valid ciphertexts can vary a lot: A pirate decoder that can decrypt ciphertexts of size lower than, say 1 GB, is *highly* imperfect, but still useful to pirates.

In this paper, we show that some traitor tracing schemes and trace and revoke schemes (including the NNL scheme from [18] and code based schemes) are susceptible to Pirates 2.0 attacks. We give several practical attacks against various instances of such schemes, most notably against the AACS. We then derive the theoretical implications for all these traitor tracing schemes.

1.2 Comparing Pirates 2.0 and the Classical Setting

We summarize below the main differences between our new attack model and the classical one:

Motivation: The classical model for coalitions captures the fact that pirates might invest some amount of money in order to sell unauthorized decoder to the black market. In the case of Pirates 2.0, the motivation might be to get rid of a protection system to which a large number of users are hostile. In the history of DVDs for instance, the main motivation to crack the system came from compatibility issues: the protection was thought to be too restrictive.

Static vs. Adaptive: The classical model of pirates is static. The coalitions consist of randomly chosen decoders. Therefore it is not possible to bias the collection process. In a Pirates 2.0 attack, traitors are able to contribute information adaptively, that is, depending on the current state of affair at the moment of the contribution. Therefore, even if during the publication process each traitor operates isolated (i.e. without communication with the other traitors), having access to published information at the time of the contribution makes it a collaborative process.

Anonymity: In the classical model of coalitions, traitors colluding must trust each other, or at least, one third party (say the pirate who collects the secret data). In contrast, the Pirates 2.0 attack only requires that the partial secret information provided by the traitors guarantees their anonymity.

Size of Coalitions: In the classical models, one usually assumes a small number of traitors (especially for combinatorial schemes like those relying on codes or those based on trees). This assumption seems reasonable in the classical model because each traitor must trust a third party and even in the case of an isolated traitor, getting a large number of decoder legally might be very expensive. In Pirates 2.0, this assumption becomes wrong, since the traitors guaranteed to remain anonymous can form a very large coalition.

2 Formalization of Pirates 2.0

There are many possible settings for a Pirates 2.0 attack. For instance, the construction of a pirate decoder can be active or passive. In the active case, the contributions made by the traitors are driven by the pirate upon building the pirate decoder. In the passive case, the traitors contribute information at their discretion. In this work, we focus on the last of these scenarios which leaves more freedom to the traitors and makes the attack even more realistic.

Also, there are two possible ways of collecting the information contributed by the traitors: in a centralized way or in a distributed way. Again, the distributed way leads to a stronger attack with less constraints in practice: traitors can easily use peer-to-peer networks to contribute their information, whereas a centralized server is more susceptible to shut down by legal action. We therefore choose to focus on the distributed setting, though in some cases, assuming a centralized entity like a pirate server would render the work of contributing for the traitors and of building a pirate decoder easier than in a peer-to-peer network. In the rest of the paper, we point out where it is relevant to use the facilities that a pirate server would provide.

2.1 A Setting for Pirates 2.0

We now describe several concepts that we use in the Pirates 2.0 setting:

Traitors and Pirates. As usual, a traitor is a legitimate user in possession of some secret data that we call his secret key and who leaks part of this secret key. Pirates are not legitimate users: they are not entitled to secret data but are able

to collect relevant information from their public environment in order to produce a pirate decoder. We naturally assume that pirates and traitors respectively collect and contribute information in a stateful way: a traitor keeps track of (all) the information he contributed to the public, whereas both pirates and traitors can keep track of all of the information that was contributed to the public.

Contributed Information. The contributed information is the sum of information that was put into the public domain by the traitors at a given point in time, i.e. the secret data leaked from the system. The current contributed information at any point in time is denoted by $\mathcal{C}$. Initially, $\mathcal{C} = \emptyset$.

Traitor's Strategy. A traitor's strategy is a publicly available probabilistic algorithm $\mathsf{Contribute}$ that traitors execute to provide information to pirates. A traitor's strategy comes with a certificate that information leaked following this strategy allows the traitors to preserve some anonymity level. Traitors might in principle use different strategies, but for simplicity we only consider in the following the case where all traitors implement the same strategy.

The strategy $\mathsf{Contribute}$, takes as input the traitor's secret key sk, some information I already contributed by other traitors (for instance the set $\mathcal{C}$ of all contributed information at the time $\mathsf{Contribute}$ is run) as well as the history H of the contributions made by the traitor. The traitor's strategy returns $\mathsf{Contribute}(sk, I, H)$ as the traitor's contribution to the public. (And therefore, the overall information contributed to the public $\mathcal{C}$ is accordingly updated: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathsf{Contribute}(sk, I, H)$.)

Public Information. The public information $\mathcal{P}$ consists of all the public data available from the broadcaster (such as for instance its public key, the public key of users if any, etc.) together with the contributed information $\mathcal{C}$.

Anonymity Level. The public procedure $\mathsf{Anonymity}$ provides the level of anonymity $\mathsf{Anonymity}(sk, S, \mathcal{P})$ of a traitor with the secret key sk who leaked an information S (which corresponds to the sum of all his contributions) following a public strategy (we refine this notion later on by using extraction functions). The anonymity level output by the procedure corresponds to the uncertainty on the traitor's identity from the tracing authority point of view when provided with the sequence of contributed information S. At level 1 the traitor is known, while at level N, the traitor is undistinguishable from another user.

Pirate Decoder. We think of a pirate decoder as the output of an algorithm called Pirate. If the amount of information available from $\mathcal{P}$ is large enough, Pirate produces a pirate decoder $\mathsf{Pirate}(\mathcal{P})$ and simply outputs '`failed`' otherwise.

In the following we assume that the contribution of secret data to the public domain $\mathcal{C}$ by the traitors is a discrete process.

Definition 1 (Security against Pirates 2.0). *A traitor tracing scheme is said to be $\alpha-$secure against Pirates 2.0 if it prevents the construction of pirate decoders from information published by traitors with an anonymity level greater than α.*

Note that not all traitor tracing (or trace and revoke) schemes are susceptible to Pirates 2.0 attacks. On the other hand, even fully collusion resistant schemes

might be at risk as Pirates 2.0 attacks allow highly imperfect decoders: decoder can refuse to decrypt classes of specific ciphertexts—e.g. depending on their size. As we will show in the next sections, some of the most famous schemes, including the one used in the AACS, are susceptible to our new attack strategy.

2.2 A Concrete Treatment of Anonymity Estimation

The basic idea behind Pirates 2.0 attacks is that traitors are free to contribute some piece of secret data as long as several users of the system could have contributed exactly the same information *following the same (public) strategy*: this way, they are able to remain somewhat anonymous. The anonymity level is meant to measure exactly how anonymous they remain.

Definition 2 (Extraction Function). *An extraction function is an efficiently computable function f that outputs information about the secret key.*

Definition 3 (Masked Traitor). *A traitor t is said to be masked by a user u for an extraction function f if $f(sk_u) = f(sk_t)$.*

This notion of a traitor being masked by another user in the system is the basic undistinguishability notion that allows us to estimate the level of anonymity of a traitor after his contribution:

Definition 4 (Anonymity Level). *The level of anonymity of a traitor t after a contribution $\cup_{1\leq i\leq n} f_i(sk_t)$ is defined as the number α of users masking t for each of the n extraction functions f_i simultaneously:*

$$\alpha = \#\{u \mid \forall i,\ f_i(sk_t) = f_i(sk_u)\} \ .$$

In the previous definitions, we use the equality between each extraction function f_i to derive the anonymity level. One can wonder why not simply consider equality between the *global* information leaked by a traitor and the global information another user u could extract like $\cup_i f_i(sk_t) = \cup_j g_j(sk_u)$ with any set of extraction functions $\{g_j\}$. The answer is that we do not want to keep the traitor strategy secret and therefore, the authority can, at least from a theoretical point of view, use its knowledge of the set of extraction functions $\{f_i\}$ used by the traitors to gain additional information and to trace the traitors. (It might well be that there exists another user u such that $\cup_i f_i(sk_t) = \cup_j g_j(sk_u)$ holds, but $\cup_i f_i(sk_t) = \cup_i f_i(sk_v)$ would have been impossible for any user v other than t.)

3 Pirates 2.0 and the Subset-Cover Framework

The subset-cover framework proposed by Naor, Naor, and Lotspiech in [18] is a powerful tool to design efficient trace and revoke systems. It captures many previously proposed traitor tracing systems and forms the basis of the so called NNL scheme used in the content protection system for HD-DVDs known as AACS [1]. However, as we show in this section, this scheme is susceptible to our attack and we explain how to defeat the AACS system.

3.1 Brief Description of the Subset-Cover Framework

The subset-cover framework is a powerful means to capture several trace and revoke designs. It encompasses several traitor tracing schemes proposed to date and maybe even more importantly, serves as the basis for two of the most efficient trace and revoke schemes: the complete subtree scheme and the subset difference scheme.

In the subset-cover framework, the set $\mathtt{N}$ of users in the system is covered by a collection of subsets $\mathtt{S}_i$ such that $\cup_i \mathtt{S}_i \supset \mathtt{N}$ and $\mathtt{S}_i \cap \mathtt{N} \neq \emptyset$. This covering is not a partition of $\mathtt{N}$ and the sets $\mathtt{S}_i$ rather overlap. To every subset $\mathtt{S}_i$ corresponds a long term secret key L_i, and every user that belongs to $\mathtt{S}_i$ is provided with this secret key—or in an equivalent way, with some material that allows him to derive this secret key. Therefore, every user u of the system is given a collection of long term keys $\{L_{i_k}\}$ that together form his secret key which we denote by sk_u.

In order to broadcast some content M, the center uses a standard hybrid scheme: a session key K is first drawn randomly and used to encrypt (with an encryption scheme E') the content, before being encrypted under multiple long term keys (with another encryption scheme E). The long term keys L_{i_k}, $k = 1, \ldots, l$ are chosen so that the corresponding subsets $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_l}$ only cover the set of users entitled to decrypt. Therefore, the center broadcasts ciphertexts of the form:

$$\Big[\big(i_1, E_{L_{i_1}}(K)\big), \big(i_2, E_{L_{i_2}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big) \parallel E'_K(M) \Big]$$

To decrypt, a valid decoder for user u performs the following sequence of operations: It first looks for an index i_j in the first element of each of the l couples $(i_k, E_{i_k}(K))$ in turn such that $\mathtt{S}_{i_j} \subset sk_u$. If no index correspond, the decoder does not decrypt; otherwise, the decoder retrieves the corresponding long term key L_{i_j} and uses it to decrypt the associated encrypted session key $E_{i_j}(K)$ and then decrypts the payload $E'_K(M)$.

Since the system is built to handle revoked users, let us also denote by $\mathtt{R}$ the set of revoked users in the system at any point in time. In order to prevent them (independently, but also working together as a coalition) from accessing the encrypted content $E'_K(M)$, the collection $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_l}$ is specially crafted so that:

$$\bigcup_{k=1}^{l} \mathtt{S}_{i_k} = \mathtt{N} \setminus \mathtt{R} \ .$$

The tracing procedure. Now that we showed how the system deals with revoked users, we have to describe the way it disables pirate decoders. As is usual, the tracing procedure works with black-box access to the pirate decoder only. The idea is to refine the covering initially used to broadcast ciphertexts so that the pirate decoder cannot decrypt with probability p higher than some threshold. To this end, the authors of [18] suggest to use an hybrid argument: the pirate box is provided with "ciphertexts" with payload $E'_K(M)$ and headers of type j (for $j = 1, \ldots, l$):

$$\big(i_1, E_{L_{i_1}}(R)\big), \ldots, \big(i_j, E_{L_{i_j}}(R)\big), \big(i_{j+1}, E_{L_{i_{j+1}}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big)$$

where R is some randomly chosen element independent from K. If we denote by p_j the probability that the pirate box correctly decrypts the specially crafted ciphertexts of type j, there must exist an index t such that $|p_t - p_{t-1}| \geq \frac{p}{l}$ and therefore some traitor belongs to $\mathtt{S}_{i_t}$. The tracer then iterates this basic procedure, applying it to an arbitrary covering of $\mathtt{S}_{i_t}$ until either $\mathtt{S}_{i_t}$ contains a single element (which thus matches a traitor) or the pirate box cannot decrypt above the threshold (and no one is accused of being a traitor, but the new partition renders the pirate box useless).

The authors of [18] showed that this tracing procedure is correct as soon as the revocation scheme satisfies a so-called "bifurcation property": every subset can be split into two subsets of roughly the same size. As we will see, this is the case for the two schemes *complete subtree* and *subset difference*.

3.2 General Attack Strategy against Subset-Cover Schemes

The generic process for the attack is relatively simple and runs in a few steps:

Elaborating the strategy

The main idea is to select a collection of subsets $\mathtt{S}_{\iota_1}, \ldots, \mathtt{S}_{\iota_w}$ such that:

- The number of users in each subset $\mathtt{S}_{\iota_k}$ is large, so that the anonymity level of the traitors is guaranteed to remain high enough when they contribute the associated long term key L_{ι_k};
- For any set $\mathtt{R}$ of revoked users and any method used by the broadcaster to partition $\mathtt{N} \setminus \mathtt{R}$ into subsets $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_m}$, the probability that one of the subsets $\mathtt{S}_{\iota_k}$ belongs to the partition $\mathtt{S}_{i_1}, \ldots, \mathtt{S}_{i_m}$ is high—say exceeds a given threshold τ—or the broadcaster exceeds its available bandwidth.

Contributing data

Let us define the extraction functions f_i to be $f_i(sk) = L_i$ if $L_i \in sk$ and '`missing`' otherwise. To contribute part of his private key sk_t, a traitor t performs the following sequence of lookups: for each index i from $\{\iota_1, \iota_2, \ldots, \iota_w\}$ (taken in any order) the traitor computes $C = f_i(sk_t)$ and if $C \neq \mathtt{failed}$ and $C \notin \mathcal{P}$ returns and outputs C. The information H about sk_t that the traitor already contributed to the public is included in the argument list so that the contribution is $\mathsf{Contribute}(sk_t, \mathcal{P}, H)$.

Building pirate decoders

A pirate decoder simply embeds the public keys $L_{\iota_1}, \ldots, L_{\iota_w}$. Upon reception of a ciphertext

$$\Big[\big(i_1, E_{L_{i_1}}(K)\big), \big(i_2, E_{L_{i_2}}(K)\big), \ldots, \big(i_l, E_{L_{i_l}}(K)\big) \parallel E'_K(M) \Big]$$

from the center, the pirate checks whether $\{\iota_1, \ldots, \iota_w\} \cap \{i_1, \ldots, i_m\} = \emptyset$. If not, that is if there is an index $\iota_k = i_l$ in both sets (which was assumed to occur with high probability), the pirate box recovers the corresponding key L_{ι_k}, uses it to decrypt the session key K from $E_{L_{i_l}}(K)$, and therefore is able to correctly decrypt the payload.

Anonymity

The level of anonymity of a given traitor t in a subset cover scheme is related to the number of users of the system that know the complete list of subsets $\mathtt{S}_{t_1}, \ldots, \mathtt{S}_{t_l}$ for which the traitor contributed the keys $L_{t_1}, \ldots, L_{t_l}$ to the public.

3.3 Pirates 2.0 against the Complete Subtree Scheme

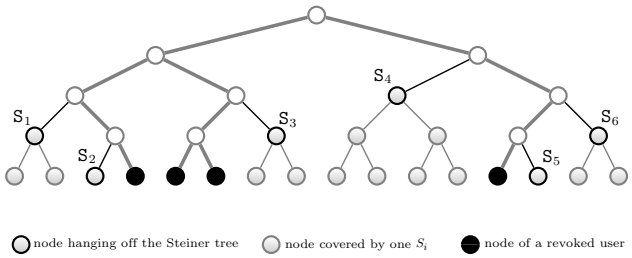


Fig. 1. Complete subtree: leaves correspond to users, $\mathtt{S}_1, \ldots, \mathtt{S}_6$ is the covering that excludes revoked users in black while allowing other users to decrypt derived from the Steiner tree associated to the set of revoked users $\mathtt{R}$.

The complete subtree scheme. In this scheme, the users correspond to the leaves of a complete binary tree whereas the collection of subsets $\mathtt{S}_i$ exactly corresponds to all the possible subtrees in the complete tree. When $|\mathtt{N}| = 2^n$, the complete binary tree is of length n and there are exactly n subtrees that contain a given leaf. Figure 1 shows a covering using six subsets of twelve users that excludes four revoked users (depicted in black). This subset scheme complies with the bifurcation property since any subset (or equivalently any subtree of the complete binary tree) can be split into two subsets of equal size (the two subtrees rooted at the two children of the root of the original subtree). Regarding key assignment, each user represented by a leaf u in the complete binary tree is provided with the keys L_i associated to the nodes i on the path from the leaf u to the root.

Covering algorithm. In the case of the complete subtree, the covering used to exclude the $r = |\mathtt{R}|$ revoked users from $\mathtt{N}$ is the collection of subsets that hang off the Steiner tree of the revoked leaves. (The Steiner tree of the revoked leaves is the minimal subtree of the complete binary tree that connects all the revoked leaves to the root and it is unique.) Since any user only knows the keys from its leaf to the root and since this path is included in the Steiner tree for revoked users, these users cannot decrypt anymore. This algorithm produces covers of size $O(r \log(N/r))$.

We now give a version of our attack against subset cover schemes in the case of the complete subtree scheme:

Theorem 1. *On average, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to mount a Pirates 2.0 attack against a complete subtree scheme in which the center wants to ensure a ciphertext rate*[3] *of at most $\rho(N-r)/N$. Moreover, each traitor is guaranteed an anonymity level of N/ρ.*

Proof. For simplicity we assume that no collision occurs during the contribution process (the traitors contribute sequentially, although in a completely random way, their share of secret data) and that the contribution of a traitor is readily available to the public. (It is obviously possible to deal with these refinements by considering statistical processes instead and then bounding the loss in efficiency that would occur in such a general case.)

Following the general attack strategy described in the previous section, define $\mathtt{S}_{\iota_1}, \ldots, \mathtt{S}_{\iota_w}$ to be the subsets corresponding to all the subtrees of the complete tree having more than N/ρ leaves so that for each ι_k more than N/ρ users share the corresponding long term keys L_{ι_k}. These subsets also correspond to all the nodes between level 0 (the root) and the level $\lambda = \lfloor \log \rho \rfloor$ and thus, there are $w = 2\lfloor \rho \rfloor$ of them. Then, a traitor contributing one of the L_{ι_k} at level λ together with every L_{ι_j} on the path from node ι_k to the root has a level of anonymity[4] higher than N/ρ. (As mentioned above, more than N/ρ users share the key L_{ι_k} and moreover the same users also know about L_{ι_i} for every node ι_i on the path from node ι_k to the root because of the assignment scheme.) Now, the number of traitors needed to collect the $\lfloor \rho \rfloor$ long term keys (and those above) is given by the answer to the classical coupon collection problem: to collect all the m possible items when one receives a uniformly chosen item at each draw requires $m \log m$ draws on average. This demonstrates the first part of the theorem.

It only remains to show that either a pirate is able to produce a working decoder, or the center uses too much bandwidth (the ciphertext rate is bigger than ρ). Let r be the number of revoked users. Let us assume that the broadcaster only uses subsets rooted at a level $l \geq \lambda$ since otherwise the priate decoder is able to decrypt the ciphertexts. Now every subset can cover at most $N/2^{\lambda}$ users so that $\rho(N-r)/N$ of them are needed to cover the $N-r$ legitimate users. □

Theoretical and practical impact. From a theoretical point of view, Theorem 1 shows that instead of the $O(r \log(N/r))$ complexity that was first derived, the bandwidth required for the complete subtree scheme to operate securely actually is $O(\rho(N-r)/N + r \log(N/r))$ for a number of $\rho \log \rho$ traitors taking part in a Pirates 2.0 attack.

From a practical point of view, we note that we assumed that every long term key can be leaked by at least one traitor. For a system accommodating 2^{32} users and a long term key at the 12th level, this assumption translates into the fact that among a million of users there is at least one that takes the step

[3] the ciphertext rate is the number of subsets used by the center

[4] having a lot of revoked users in the subtree does not affect the level of anonymity: revoked users know the keys on their path to the root and could have contributed them as well. This, however, affects the decryption threshold of the pirate decoder

of contributing it to the public (with the guarantee of remaining anonymous!); this hypothesis seems reasonable to us.

Also, note that even in the case where one long term key is not contributed by any user, the attack remains valid: the pirate box will not be able to decrypt only with a very small probability.

3.4 Pirates 2.0 against the Subset Difference Scheme

The subset difference scheme has been introduced to lower the number of subsets required to partition the set of legitimate users $\mathtt{N} \setminus \mathtt{R}$. It improves on the complete subtree scheme exposed above by a factor of $\log(N/r)$ in terms of bandwidth usage for the headers.

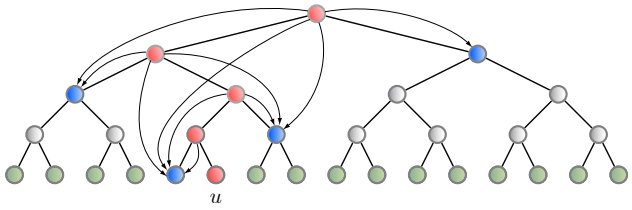


Fig. 2. Key assignment. User u receives all the labels $\mathsf{LABEL}_{i,j}$ such that i is a parent of j and i is on the path from the leaf of u to the root.

To attain this level of performance, the number of possible subsets has been tremendously increased. Remember that $\mathtt{S}_i$ denotes the full binary subtree of the complete binary tree rooted at node i. Now, for each node j in $\mathtt{S}_i$ different from i, let us denote by $\mathtt{S}_{i,j}$ the binary subtree rooted at node i of which the full binary subtree rooted at node j has been removed. (See examples in Figure 3.) A user will need to know all the keys $L_{i,j}$ such that he belongs to the subtree rooted at i but not in the subtree rooted at j. However, it would be impossible for each device to store such a huge number of long term keys. This is why a key derivation procedure has been designed to allow the derivation of most of the $O(N)$ long term keys a user is entitled from a much smaller set of keys: a user only needs to store $O(\log^2(N))$ keys. Each node i in the full binary tree is first assigned a random label LABEL_i and labels $\mathsf{LABEL}_{i,j}$ together with their corresponding long term keys $L_{i,j}$ are deduced (in a pseudo-random way) from label LABEL_i. The key derivation procedure then works as follows: from each LABEL_i, a pseudo-random value $\mathsf{LABEL}_{i,j}$ is obtained for each sub-node j using the tree based construction proposed by Goldreich, Goldwasser, and Micali [13]; from this value, a long term key $L_{i,j}$ is eventually deduced (in a pseudo-random way). Each user is then provided with labels $\mathsf{LABEL}_{i,j}$ for all nodes i that are on the path from the leaf that represents the user to the root, and all nodes j hanging off this path as described on Fig. 2. This key assignment ensures that every user in the subtree rooted at node i but not in the subtree rooted at node j

is able to derive $L_{i,j}$ while every user in the subtree rooted at node j is not able to derive $L_{i,j}$.

Covering algorithm. The covering algorithm works by maintaining a subtree T of the Steiner tree of $\mathtt{R}$ and removes nodes from it at each steps:

`init:` Make T the Steiner tree of $\mathtt{R}$.

`select:` If there is only one leaf v_k in T and it is not the root (or node 0), add the subset $\mathtt{S}_{0,k}$ and return. If there is only the root in T, return. Otherwise, select two leaves v_{j_1} and v_{j_2} from T so that their least common ancestor v does not contain any other leave of T than v_{j_1} and v_{j_2}. Call v_{i_1} and v_{i_2} the children of v such that v_{i_1} is the ancestor of v_{j_1} and v_{i_2} the ancestor of v_{j_2}. Then, if $v_{i_1} \neq v_{j_1}$ add $\mathtt{S}_{i_1,j_1}$ to the partition and similarly if $v_{i_2} \neq v_{j_2}$ add $\mathtt{S}_{i_2,j_2}$ to the partition. Remove all the descendants of v from T, which makes v a leaf of T. Reiterate the step 'select'.

An example output of this procedure is shown in Figure 3.

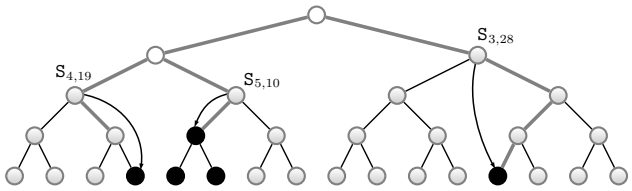
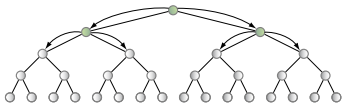


Fig. 3. Subset difference: leaves correspond to users and black nodes are not able to derive the necessary information to decrypt. Therefore $\mathtt{S}_{4,19}$ prevents user 19 from decrypting, $\mathtt{S}_{5,10}$ prevents users 20 and 21 from decrypting, and $\mathtt{S}_{3,28}$ prevents user 28 from decrypting. All other users are able to decrypt.

Theorem 2. *On average, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to mount a Pirates 2.0 attack against a subset difference scheme in which the center wants to ensure a ciphertext rate of at most $\rho(N-r)/N$. Moreover, each traitor is guaranteed a level of anonymity of at least $N/2\rho$.*

Proof. In the following proof we make use of labels of a special type, that we call *direct labels*. Direct labels are $\mathsf{LABEL}_{i,j}$ such that the node j is a *direct* descendant of the node i. The first six direct labels of the tree are described in the figure on the left.



First, note that a pirate knowing all the keys $L_{i,j}$ where the node i lies in the first $\lambda = \lfloor \log \frac{\rho}{2} \rfloor$ levels, is able to decrypt all the ciphertexts where the rate is lower than $\rho(N-r)/N$ where r is the number of revoked users. Indeed, the broadcaster must use subsets $S_{k,l}$ where the node k does not lie in the first λ levels in order to prevent the pirate from decrypting the ciphertexts. Since

each of these subsets covers less than $N/2^{\lambda+1}$ users (those who are in the subtree rooted at node k), the center must use at least $\rho(N-r)/N$ subsets to cover all the legitimate users.

Collecting all the keys rooted at a level $l \leq \lambda$ is however totally unpractical since there are a tremendous number of such keys. The pirate can nevertheless go around this difficulty by collecting labels $\mathsf{LABEL}_{i,j}$ instead of keys $L_{i,j}$ and using the derivation procedure to lower the minimum information to be kept: the labels that users possess allow to derive a large number of keys. Therefore, we claim that it is enough for the pirate to collect all *direct* labels $\mathsf{LABEL}_{i,j}$ where i is located in the first λ levels in order to derive all keys $L_{i,k}$. (Once the pirate knows the two direct labels at node i, he can derive all keys $L_{i,k}$ where k is in the subtree rooted at i.)

To prove the theorem, we show that on average, $\rho \log \rho$ randomly chosen traitors are able to contribute all the direct labels of the first λ levels. Each traitor contributes all his direct labels $\mathsf{LABEL}_{i,j}$ for the nodes i located in the first λ levels. Note that at each level, a traitor has been assigned exactly one of the direct labels. Thus, when all direct labels at level exactly λ have been contributed, so have the direct labels of all the first $\lambda - 1$ levels. As a randomly chosen traitor knows a uniformly chosen direct label out of the $\frac{\rho}{2}$ direct labels of level λ, a randomly chosen group of $\rho \log \rho$ traitors (operating isolated) is able to contribute all direct labels $\mathsf{LABEL}_{i,j}$ where i is located in the first λ levels.

Moreover, such traitors share their contribution with every user in the same subtree rooted at level $\lambda + 1$: each traitor is covered by N/ρ users. □

Remark 1. Theorem 2 is proven in the case of static attacks: traitors submit information non-adaptively, such as in a peer-to-peer scenario. However, the number of required traitors to mount a Pirate 2.0 attack can be lowered to ρ in the case of an adaptive attack such as in a server-based scenario.

Impact on AACS. In the case of AACS, the subset difference scheme is used with $N = 2^{31}$ users. The header is written in a so-called Media Key Block or MKB for short which (among other) encodes the indices for the difference subsets as well as the media key encrypted once for each of the corresponding long term keys. These keys are 16 bytes long and the indices are encoded using 5 bytes. According to Section 3.2.5.5 of AACS specifications [2]: "*For the purposes of designing performance, a one megabyte buffer is sufficient to process the MKB.*" Although this is not an intrinsic limitation of the system, very large MKBs would decrease the performances of hardware devices and would increase their price. This is why applications like disk replicators often only allocate 1MB space for the MKB. In the case of AACS, this means that only $2^{11.6} = 2^{20}/21$ encrypted keys will be able to fit this space and thus a Pirates 2.0 attack against the AACS would only require some thousand collaborating traitors which, given the guarantee offered to traitors (a million of other users cover each traitor), seems very practicable.

Also note that once again the attack given here is just an illustration of our general concept of attack. There are several possible improvements and refinements such as taking advantage of the partition algorithm (remember that the

scheme is a trace and revoke scheme and not a full traitor tracing scheme, so that it might fail to single out a traitor).

4 Pirates 2.0 and Code Based Schemes

Traitor tracing schemes based on codes (be it collusion secure codes [7, 25] or identifiable parent property codes [14, 22]) have been proposed during more than half a decade [17, 8, 21, 20, 23, 5]. Their main advantage is their efficiency in terms of bandwidth requirements, but their main drawback is that their efficiency (in terms of the size of the private key) is highly sensitive to the number of traitors in the coalition.

4.1 General Framework of Codes Based Schemes

Traitor tracing schemes built on codes more or less fit in the following framework:

Setup: The scheme generates a code $\mathtt{C}$ of length ℓ which is either a collusion secure code or a q-ary c-IPP code. The alphabet for the code is $\mathtt{A} = \{0, 1\}$ in the case of a collusion secure code and $\mathtt{A} = \{1, \ldots, q\}$ in the case of an IPP code. Then, for each position $i = 1, \ldots, \ell$ in a codeword and for each possible letter a from $\mathtt{A}$, a key $K_{i,a}$ is randomly chosen. Hence, there are 2ℓ possible keys (resp. $q\ell$ possible keys) in the system in the case of collusion secure codes (resp. IPP codes).

Key assignment: Each user u is given a codeword W_u from $\mathtt{C}$. Then, for each position $i = 1, \ldots, \ell$ in this codeword, the user is provided with the key $K_{i,W_u[i]}$ where $W_u[i]$ is the letter at position i in the codeword W_u. Thus, each user gets ℓ keys in its decoder.

Decoder: A ciphertext usually contains a header that specifies the positions of the keys involved in the decryption process. For instance, in the case of the scheme [17] proposed by Kiayias and Yung, all the keys of the user are involved. In the case of the scheme [5] proposed by Boneh and Naor only one key is involved during a decryption process.

4.2 Pirates 2.0 against Code Based Schemes

Our goal is to show how our generic attack can be applied to this class of schemes. We do not focus on any concrete construction but rather deal with the underlying codewords. For ease of exposition, we describe an attack when the underlying code is a Tardos' code [25] but this attack might easily translate to other codes.

First, recall that a Tardos' code secure against coalitions of size at most c is built as follows. First, the code length is set to be $\ell = \lceil 100c^2 \log(N/\epsilon) \rceil$. Then, for each integer i in the interval $[1, \ldots, \ell]$ a (secret) value $0 < p_i < 1$ heavily biased towards 0 or 1 is randomly drawn. Then, any of the N codewords is constructed by randomly choosing for each position i in $[1, \ldots, \ell]$ the bit '$\mathtt{0}$' or the bit '$\mathtt{1}$' according to the probability p_i.

Theorem 3. *For any traitor tracing scheme that relies on Tardos' code for its set of keys, a set of T traitors collaborating to mount a Pirates 2.0 attack allows to produce a pirate decoder while maintaining a level of anonymity higher than $N \cdot 2^{-\ell/T}$ on the average.*

Proof. Since contributing large amounts of a codeword makes your level of anonymity drop a lot, a strategy that handles every traitor with equity is to make them contribute the same amount of secret data. Since there are T traitors, let them each contribute ℓ/T elements of (the secret data associated with) their codeword. Of course, people are then already able to construct pirate decoders with the collected material. The anonymity level α a traitor can expect is easy to assess: if $m = \lceil \ell/T \rceil$,

$$\alpha = N \prod_{i=1}^{m} \left(p_{\sigma(i)}^2 + (1 - p_{\sigma(i)})^2 \right) \quad . \tag{1}$$

Indeed, for a randomly chosen traitor, there is a probability p_i that the letter at position i is '0' and for any other codeword randomly chosen a probability p_i that the letter at that position is also '0'. Similarly there is a probability $1-p_i$ that the letter at position i is '1' and the same probability that another codeword gets the same letter at that position. Therefore, the probability that another codeword gets the same letter as that of the traitor for some position i is $q_i = p_i^2 + (1-p_i)^2$. The probability that a block of size m of the traitor's codeword is the same as that of another user is thus $\prod_{i=1}^{m} q_{\sigma(i)}$, where σ is a permutation of $\{1, \ldots, \ell\}$ that accounts for the particular selection of the block of size m.

The sum from Eq. (1) takes into account every possible block of codeword of length m and by multiplying by the total number of users in the system, we get the average number of users masking a randomly chosen traitor, that is its level of anonymity in the system. Now since $p_i^2 + (1 - p_i)^2 \geq \frac{1}{2}$ we get a (very loose) bound on the level of anonymity: $\alpha \geq N \cdot 2^{-\ell/T}$. □

Theoretical and practical impact. From a theoretical point of view, the above theorem shows that the number of traitors required to mount a Pirates 2.0 is only *linear in the size of the decoder* and only logarithmic in the number of users in the system. From a practical point of view, it would require about 2^{17} traitors to mount a Pirate 2.0 attack against a traitor tracing scheme that relies on a 30-collusion secure code with 2^{32} users. Each traitor would be masked by about a few thousand users in this case.

5 Conclusion

Throughout this paper we presented a novel concept of attack against combinatorial traitor tracing schemes. We focused on the main ideas behind this concept of attack, but some variations could be further investigated. For instance, it is possible to consider the case of dishonest traitors (a common threat to collaborative work is bad contributions which have to be tracked and eliminated).

Dishonest traitors capture the fact that the authority could try to perturb the creation of pirate decoders by publishing incorrect information. However, one of the traitors might use its own authorized decoder to verify the contribution of the other traitors: after having sorted out these contributions, he is able to produce a pirate decoder.

Another direction is to consider probabilistic guarantees for the level of anonymity of contributing traitors: the traitors are only certified to have a high level of anonymity with some (possibly very high) probability. This is useful if the authority tries to embed markers specific to a single user. However, there is a trade-off for the authority between the effectiveness of this process against Pirates 2.0 and the efficiency of the scheme.

Eventually, the most interesting direction is probably to provide modified versions of the common traitor tracing schemes that resist Pirates 2.0 attacks without sacrificing the efficiency of the original schemes.

References

1. AACS LA. AACS Specifications. At `http://www.aacsla.com/specifications/`.
2. AACS LA. Introduction and Common Cryptographic Elements. Downloaded from: `http://www.aacsla.com/specifications/specs091/AACS_Spec_Common_0.91.pdf`.
3. Olivier Billet and Duong Hieu Phan. Efficient Traitor Tracing from Collusion Secure Codes. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security—ICITS 2008*, volume 5155 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2008.
4. 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.
5. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertexts, 2008.
6. 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.
7. 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.
8. Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public Traceability in Traitor Tracing Schemes. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 542–558. Springer, 2005.
9. 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.
10. DefectiveByDesign. `http://www.defectivebydesign.org/`.
11. Electronic Frontier Foundation. `http://www.eff.org/`.
12. 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.

13. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *Symposium on Foundations of Computer Science—FOCS 84*, pages 464–479. IEEE, 1984.
14. Henk D. L. Hollmann, Jack H. van Lint, Jean-Paul M. G. Linnartz, and Ludo M. G. M. Tolhuizen. On Codes with the Identifiable Parent Property. *J. Comb. Theory, Ser. A*, 82(2):121–133, 1998.
15. Aggelos Kiayias and Serdar Pehlivanoglu. Pirate Evolution: How to Make the Most of Your Traitor Keys. In Alfred Menezes, editor, *Advances in Cryptology—CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 448–465. Springer, 2007.
16. Aggelos Kiayias and Moti Yung. On Crafty Pirates and Foxy Tracers. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management—DRM 2001*, volume 2320 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2002.
17. 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.
18. 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.
19. Birgit Pfitzmann. Trials of Traced Traitors. In Ross J. Anderson, editor, *Information Hiding—IH '96*, volume 1174 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 1996.
20. 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.
21. 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.
22. Palash Sarkar and Douglas R. Stinson. Frameproof and IPP Codes. In C. Pandu Rangan and Cunsheng Ding, editors, *Progress in Cryptology—INDOCRYPT 2001*, volume 2247 of *Lecture Notes in Computer Science*, pages 117–126. Springer, 2001.
23. 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.
24. Stop DRM Now! `http://stopdrmnow.org/`.
25. Gábor Tardos. Optimal probabilistic fingerprint codes. In *ACM Symposium on Theory of Computing—STOC 2003*, pages 116–125. ACM, 2003.