Pirate Evolution: How to make the most of your traitor keys

Aggelos Kiayias* and Serdar Pehlivanoglu*

Computer Science and Engineering, University of Connecticut
Storrs, CT, USA. {aggelos,sep05009}@cse.uconn.edu

Abstract. We introduce a novel attack concept against trace and revoke schemes called pirate evolution. In this setting, the attacker, called an evolving pirate, is handed a number of traitor keys and produces a number of generations of pirate decoders that are successively disabled by the trace and revoke system. A trace and revoke scheme is susceptible to pirate evolution when the number of decoders that the evolving pirate produces exceeds the number of traitor keys that were at his possession. Pirate evolution can threaten trace and revoke schemes even in cases where both the revocation and traceability properties are ideally satisfied: this is because pirate evolution may enable an attacker to "magnify" an initial key-leakage incident and exploit the traitor keys available to him to produce a great number of pirate boxes that will take a long time to disable. Even moderately successful pirate evolution affects the economics of deployment for a trace and revoke system and thus it is important that it is quantified prior to deployment.

In this work, we formalize the concept of pirate evolution and we demonstrate the susceptibility of the trace and revoke schemes of Naor, Naor and Lotspiech (NNL) from Crypto 2001 to an evolving pirate that can produce up to $t \cdot \log N$ generations of pirate decoders given an initial set of t traitor keys. This is particularly important in the context of AACS, the new standard for high definition DVDs (HD-DVD and Blue-Ray) that employ the subset difference method of NNL: for example using our attack strategy, a pirate can potentially produce more than 300 pirate decoder generations by using only 10 traitor keys, i.e., key-leakage incidents in AACS can be substantially magnified.

1 Introduction

A trace and revoke scheme is an encryption scheme that is suitable for digital content distribution to a large set of receivers. In such a scheme, every receiver possesses a decryption key that is capable of inverting the content scrambling mechanism. The defining characteristics of a trace and revoke scheme are the following: (i) *revocation*: the sender can scramble content with a "broadcast pattern" in such a way so that the decryption capability of any subset of the receiver population can be disabled, (ii) *tracing*: given a rogue decryption device (called a

* Research partly supported by NSF CAREER Award CNS-0447808.

pirate decoder) that was produced using the keys of a number of receivers (called traitors) it is possible to render such device useless from future transmissions. This can be done by identifying the traitors and revoking them or in some other fashion (that may not involve the direct identification of any traitor).

Trace and revoke schemes conceptually are a combination of two cryptographic primitives that have been originally suggested and studied independently: broadcast encryption, introduced by Fiat and Naor in [10] and studied further in e.g., [13, 14, 22, 8, 15, 16, 21], and traitor tracing and related codes, introduced by Chor, Fiat and Naor in [6] and studied further in e.g., [33, 23, 20, 2, 28–30, 17–19, 31, 34, 5, 27]. The combination of the two primitives appeared first in [24] and explored further in [9]. Trace and revoke schemes for stateless receivers were proposed in [22] and explored further in [15, 16]. The stateless receiver setting is of particular interest since it does not require receivers to maintain state during the life-time of the system; this greatly simplifies the system aspects and deployment management of a trace and revoke scheme.

The security requirements for trace and revoke schemes are relatively well understood when one considers the revocation or tracing components in isolation: the revocation component should be coalition resistant to an adversary that adaptively joins the system, is entirely revoked and subsequently attempts to decrypt a ciphertext. The tracing component should also be coalition resistant: an adversary given a set of keys should be incapable of producing a pirate decoder that cannot have at least one traitor identified. When Naor, Naor and Lotspiech [22] introduced the broadcast encryption framework of subset cover, they made the nice observation that if a broadcast encryption scheme satisfies a property called “bifurcation” then it is possible to construct an efficient tracing procedure that will produce ciphertexts that are unreadable by any given rogue pirate decoder; this satisfies the requirements for a trace and revoke scheme (albeit without identifying traitors directly). They proposed two combinatorial designs (called the complete-subtree and subset-difference method) for broadcast encryption that satisfy bifurcation and thus produced two trace and revoke schemes. The subset-difference scheme is particularly attractive as it enjoys a linear communication overhead during encryption (linear in r the number of revoked users) and it was employed as the basis for the new high definition DVD encryption standard, the AACS [1].

It is common in cryptographic design when a construction combines simultaneously two security functionalities (even when they are well understood in isolation) that the possibility for new forms of attacks springs up. In our case, in a trace and revoke scheme the adversaries that have been considered so far were attacking directly the revocation component (they were revoked and attempted to evade revocation) or the traceability component (they produced a pirate decoder that attempted to evade the tracing algorithm). This raises the question, in a trace and revoke scheme, are these the only relevant attack scenarios?

Pirate Evolution. Pirate evolution is a novel attack concept against a trace and revoke scheme that exploits the properties of the combined functionality of tracing and revocation in such a scheme. In a pirate evolution attack, a pirate obtains a set of traitor keys through a "key-leaking" incident. Using this set of keys the pirate produces an initial pirate decoder. When this pirate decoder is captured and disabled by the transmission system using the tracing mechanism, the pirate "evolves" the first pirate decoder by issuing a second version that succeeds in decrypting ciphertexts that have the broadcast pattern disabling the first decoder. The same step is repeated again and the pirate continues to evolve a new version of the previous decoder whenever the current version of pirate decoder becomes disabled from the system. A pirate that behaves as above will be called an evolving pirate and each version of the pirate decoder will be called a generation (as presumably many copies of the same pirate decoder may be spread by the pirate).

This is a novel attack concept as the adversary here is not trying to evade the revocation or the traceability component. Instead he tries to remain active in the system for as long as possible in spite of the efforts of the administrators of the system. We say that a trace and revoke scheme is immune to pirate evolution if the number of generations that an evolving pirate can produce equals the number of traitor keys that have been corrupted (i.e., the number of traitors). The number of traitors is a natural lower bound to the generations that an evolving pirate can produce: trivially, an evolving pirate can set each version it releases to be equal to the decoder of one of the traitors. Nevertheless, the number of generations that a pirate may produce can be substantially larger depending on the combinatorial properties of the underlying trace and revoke system. We call the maximum number of decoders an evolving pirate can produce, the pirate evolution bound evo of the trace and revoke scheme. Note that this bound will be a function of the number of traitors t as well as of other parameters in the system (such as the number of users).

When evo is larger than t, we say that a trace and revoke scheme is susceptible to pirate evolution. When evo is much larger than t, this means that an initial leaking incident can be "magnified" and be of a scale much larger than what originally expected. Interestingly, a system may satisfy both the tracing and revocation properties in isolation and still be rendered entirely useless if evo is sufficiently large (say super-poly in t). In this case the trace and revoke scheme could be defeated by simply taking too long to catch up with an evolving pirate that could keep exploiting a minor initial key leakage incident to produce a multitude of pirate decoders in succession.

Even when evo is just moderately larger than t, it is an important consideration for a trace and revoke scheme in an actual deployment. The economics of a deployment would be affected and we believe that resilience against pirate evolution attacks should be part of the suggested considerations.

In this work, we introduce and study pirate evolution in the subset cover framework of stateless receivers of [22]. We first formalize the concept of pirate evolution through the means of an attack game played between the evolving pirate and a challenger that verifies certain properties about the pirate decoders produced by the evolving pirate. Next, we prove that it is in fact possible to design trace and revoke schemes that are immune against pirate evolution by presenting a simple design that renders any evolving pirate incapable of producing more pirate decoders than traitors. This result (albeit not very efficient as a trace and revoke scheme) shows that immunity against pirate evolution is attainable in principle. Still, it is interesting to note that immunity may come at a high cost for certain systems and thus it could be desirable in many settings to sacrifice immunity in favor of efficiency if the amount of pirate evolution that is possible is deemed to be within acceptable limits for the economics of a certain system deployment (compare this to the usual example of a bank that allows a few fraudulent transactions if the incurred losses can be factored into the profits).

Next, we focus on the complete-subtree and subset-difference trace and revoke systems of [22]. We demonstrate both these schemes are susceptible to pirate evolution. Each of our pirate evolution attacks requires careful scheduling of how the traitor keys are expended by the pirate; moreover in both cases there is sensitivity to the "geometry" of the leaking incident. For the complete subtree method we present a pirate evolution attack that given t traitor keys, it enables an evolving pirate to produce up to $t \log(N/t)$ generations where N is the total number of users in the system (actually number of leaves in the tree, as the set of currently active users may be much less). For the subset difference method we present a pirate evolution attack that given t traitor keys, it enables an evolving pirate to produce up to $t \log N$ generations of pirate decoders.

In the context of AACS, [1], the new encryption standard for high definition DVDs, where the subset-difference method of [22] is deployed, the pirate evolution attack we present suggests that each single traitor key can be used to yield up to 31 generations of pirate decoders (refer to section 3.3).

2 The Subset-Cover Revocation Framework

The Subset-Cover revocation framework [22] is an abstraction that can be used to formulate a variety of revocation methods. It defines a set of subsets that cover the whole user population and assigns (long-lived) keys to each subset; each user receives a collection of such keys (or derived keys). We denote by N the set of all users where $|\mathsf{N}| = N$ and $\mathsf{R} \subset \mathsf{N}$ the set of users that are to be revoked at a certain instance where $|\mathsf{R}| = r$. Note that N is not necessarily the set of currently active users but the number of all users that are anticipated in the lifetime of the system.

The goal of the sender is to transmit a message M to all users such that any $u \in \mathsf{N} \setminus \mathsf{R}$ can recover the message whereas the revoked users in R can not recover it. Note that the non-recovery property should also extend to any coalition of revoked users. The framework is based on a collection of subsets $\{\mathsf{S}_j\}_{j \in \mathcal{J}}$ where $\mathsf{S}_j \subseteq \mathsf{N}$ such that any subset $\mathsf{S} \subseteq \mathsf{N}$ can be partitioned into disjoint subsets of $\{\mathsf{S}_j\}_{j \in \mathcal{J}}$. Each subset S_j is associated with a *long-lived* key $\mathtt{L}_j$. Users are assumed to be initialized privately with a set of keys such that u has access to $\mathtt{L}_j$ if and only if $u \in \mathsf{S}_j$. The private data assigned to user u in this intialization step will be denoted by $\mathcal{I}_u$. In particular we define $\mathcal{I}_u = \{j \in \mathcal{J} \mid u \in \mathsf{S}_j\}$

and $\mathcal{K}_u = \{\mathsf{L}_j \mid j \in \mathcal{I}_u\}$. Given a revoked set R, the remaining users $\mathsf{N} \setminus \mathsf{R}$ are partitioned into disjoint $\{\mathsf{S}_{i_1}, \ldots, \mathsf{S}_{i_m}\} \subset \{\mathsf{S}_j\}_{j \in \mathcal{J}}$ so that $\mathsf{N} \setminus \mathsf{R} = \bigcup_{j=1}^{m} \mathsf{S}_{i_j}$. The transmission of the message M is done in a hybrid fashion. First a random session key K is encrypted with all *long-lived* keys $\mathtt{L}_{i_j}$ corresponding to the partition, and the message M is encrypted with the session key. Two encryption functions are being used in this framework: (1) $\mathcal{F}_K : \{0,1\}^* \mapsto \{0,1\}^*$ to encrypt the message. (2) $\mathcal{Q}_L : \{0,1\}^l \mapsto \{0,1\}^l$ to encrypt the session key. Each broadcast ciphertext will have the following form:

$$\langle \underbrace{[i_1, i_2, \ldots i_m, \mathcal{Q}_{L_{i_1}}(K), \mathcal{Q}_{L_{i_2}}(K), \ldots \mathcal{Q}_{L_{i_m}}(K)]}_{\text{HEADER}}, \underbrace{\mathcal{F}_K(M)}_{\text{BODY}} \rangle \tag{1}$$

The receiver u decrypts a given ciphertext $\mathcal{C} = \langle [i_1, i_2, \ldots i_m, C_1, C_2, \ldots C_m], M' \rangle$ as follows: (i) Find i_j such that $u \in \mathsf{S}_{i_j}$, if not respond null, (ii) Obtain $\mathtt{L}_{i_j}$ from $\mathcal{K}_u$. (iii) Decrypt the session key: $K' = \mathcal{Q}^{-1}_{L_{i_j}}(C_j)$. (iv) Decrypt the message: $M = \mathcal{F}^{-1}_K(M')$. In [22], two methods in the subset cover framework are presented called the Complete Subtree `CS` and the Subset Difference `SD`.

Tracing Traitors in the Subset Cover Framework. Beyond revoking sets of users that are not supposed to receive content, trace and revoke schemes are supposed to be able to disable the rogue pirate decoders which are constructed using a set of traitor's keys that are available to the pirate. One way this can be achieved is to identify a traitor given access to a pirate box and then add him to the set of revoked users. Given that the goal of tracing is to disable the pirate box, the NNL tracing algorithm focuses on just this security goal. In the NNL setting, it is sufficient to find a "pattern" which makes the pirate box unable to decrypt.

Regarding the tracing operation, the following assumptions are used for the pirate decoder: (1) the tracing operation is black-box, i.e., it allows the tracer to examine only the outcome of the pirate decoder as an oracle. (2) the pirate decoder is not capable of recording history; (3) the pirate decoder lacks a "locking" mechanism which will prevent the tracer to pose more queries once the box detects that it is under tracing testing. (4) the pirate decoder succeeds in decoding with probability greater than or equal to a threshold q.

Based on the above, the goal of the tracing algorithm is to output *either* a non-empty subset of traitors, *or* a partition of $\mathsf{N} \setminus \mathsf{R} = \bigcup_{j=1}^{m} \mathsf{S}_{i_j}$ for the given revoked users R, such that if this partition is used to distribute content M in the framework as described above it is impossible to be decrypted by the pirate box with sufficiently high probability (larger than the threshold q); at the same time, all good users can still decrypt.

The tracing algorithm can be thought of as a repeated application of the following basic procedure that takes as input a partition: First it is tested whether the box decrypts correctly with the given partition $\bigcup_{j=1}^{m} \mathsf{S}_{i_j}$ (with probability p greater than the threshold). If not, the subset tracing outputs the partition as the output of the tracing algorithm. Otherwise, it outputs one of the subsets containing at least one of the traitors. The tracing algorithm then partitions that subset somehow and inputs the new partition (that is more "refined") to the next iteration of the basic procedure. If the subset resulting by the basic procedure contains only one possible candidate, then we can revoke that user since it is a traitor. Here is how the basic procedure works:

Let p_j be the probability that the box decodes the special tracing ciphertext

$$\langle [i_1, i_2, \ldots i_m, \mathcal{Q}_{L_{i_1}}(R), \mathcal{Q}_{L_{i_2}}(R), \ldots \mathcal{Q}_{L_{i_j}}(R), \mathcal{Q}_{L_{i_{j+1}}}(K), \ldots \mathcal{Q}_{L_{i_m}}(K)], \mathcal{F}_K(M)\rangle$$

where R is a random string of the same length as the key K. Note that $p_0 = p$ and $p_m = 0$, hence there must be some $0 < j \leq m$ for which $|p_{j-1} - p_j| \geq \frac{p}{m}$. Eventually, this leads the existence of a traitor in the subset S_{i_j} under the assumption that it is negligible to break the encryption scheme $\mathcal{Q}$ and the key assignment method.

The above can be turned into a full-fledged tracing algorithm, as long as the Subset-Cover revocation scheme satisfies the "Bifurcation property": any subset S_k can be partitioned into not extremely uneven sets S_{k_1} and S_{k_2}. Both `CS` and `SD` methods allow us to partition any subset S_k into two subsets with the Bifurcation property. For the Complete Subset, it is simply taking the subsets rooted at the children of node v_k. For the `SD` method, given $\mathsf{S}_{i,j}$ we take the subsets $\mathsf{S}_{i,c}$ and $\mathsf{S}_{c,j}$ where v_c is a child of the node v_i and v_j is on the subset rooted at v_c. Formally, we have the following definition for tracing algorithm and encryption procedure after tracing pirate boxes to disable them recovering message:

Definition 1. *For a given set of revoked users* R *and pirate boxes* $B_1, B_2, \cdots, B_s$ *caught by the sender, the encryption function first finds a partition* $\mathcal{S}$ *which renders the pirate boxes useless and outputs the ciphertext. Let* $\mathcal{T}$ *be the tracing function outputting the partition to render the pirate boxes useless, then:* $\mathcal{T}^{B_1,B_2,\cdots,B_s}(\mathsf{R}) = \mathcal{S}$. *Denote the ciphertext created by the encryption scheme interchangeably by following notations:* $\mathcal{C} = \mathcal{E}_{\mathcal{R}}^{B_1,B_2,\cdots,B_s}(M)$ or $\mathcal{C} = \mathcal{E}_{\mathcal{S}}(M)$, *where* $\mathcal{E}_k^{-1}(\mathcal{C}) = M$ *for* $k \notin \mathsf{R}$, *and any pirate box* B_i, $0 < i \leq s$, *decrypts the ciphertext with probability less than threshold* q, *i.e.* $Prob[B_i(\mathcal{C}) = M] < q$.

According to the above definition, the sender applies tracing algorithm on the pirate boxes he has access to before broadcasting the message.

3 Pirate Evolution

In this section we introduce the concept of pirate evolution. We present a game based definition that is played with the adversary which is the "evolving pirate." Let t be the number of traitor keys in the hands of the pirate. The traitor keys are made available to the pirate through a key-leaking "incident" $\mathcal{L}$ that somehow chooses a subset of size t from the set $\{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$ (the set of all users' private data assigned by a function $\mathcal{G}$ with a security parameter λ). We permit $\mathcal{L}$ to be also based on the current set of revoked users R. Specifically, if $\mathsf{T} = \mathcal{L}(\mathcal{I}_1, \mathcal{I}_2, \cdots \mathcal{I}_n, t, \mathsf{R})$ then $|\mathsf{T}| = t$, $\mathsf{T} \subseteq \{\mathcal{I}_u \mid u \in \mathsf{N} \setminus \mathsf{R}\}$. This models the fact that the evolving pirate may be able to select the users that he corrupts. Separating the evolving pirate from the leaking incident is important though as it enables us to describe how a pirate can deal with leaking incidents that are not necessarily the most favorable (the pirate evolution attacks that we will describe in the sequel will operate with any given leaking incident and there will be leaking incidents that are more favorable than others). We note that partial leaking incidents can also be considered within our framework.

Once the leaking incident determines the private user data that will be available to the evolving pirate (i.e., the traitor key material), the evolving pirate $\mathcal{P}$ receives the keys and produces a "master" pirate box $\mathcal{B}$. The pirate is allowed to have oracle access to an oracle $\mathcal{E}_{\mathsf{R}}(\mathcal{M})$ that returns ciphertexts distributed according to plaintext distribution that is employed by the digital content distribution system (i.e., the access we consider is not adaptive); an adaptive version of the definition (similar to a chosen plaintext attack against symmetric encryption) is also possible.

Given the master pirate box, an iterative process is initiated: the master pirate box spawns successively a sequence of pirate decoders $B_1, B_2, \ldots$ where $B_i = \mathcal{B}(1^{t+\log N}, \ell)$ for $\ell = 1, 2, \ldots$. Note that we loosely think that the master box is simply the compact representation of a vector of pirate boxes; the time complexity allowed for its operation is polynomial in $t + \log N + \log \ell$ (this can be generalized in other contexts if needed — we found it to be sufficient for the evolving pirates strategies we present here). Each pirate box is tested whether it decrypts correctly the plaintexts that are transmitted in the digital content distribution system with success probability at least q. The first pirate box is tested against the "initial" encryption function $\mathcal{E}_{\mathsf{R}}(\cdot)$, whereas any subsequent box is tested against $\mathcal{E}_{\mathsf{R}}^{B_1,B_2,\cdots B_{i-1}}(\cdot)$ which is the encryption that corresponds to the conjunctive revocation of the set R and the tracing of all previous pirate boxes. The iteration stops when the master pirate box $\mathcal{B}$ is incapable of producing a pirate decoder with decryption success exceeding the threshold q. Each iteration of the master box corresponds to a "generation" of pirate boxes. The number of successfully decoding pirate generations that the master box can spawn is the output of the game-based definition given below. The trace and revoke scheme is susceptible to pirate evolution if the number of generations returned by the master box is greater than t. Note that the amount of susceptibility varies with the difference between the number of generations and t; the pirate evolution bound evo is the highest number of generations any evolving pirate can produce. Formally, we have the following:

Definition 2. *Consider the game of figure 1 given two probabilistic machines $\mathcal{P}, \mathcal{L}$ and parameters $\mathsf{R} \subseteq \{1, 2, \cdots n\}$, t, $r = |\mathsf{R}|, q$. Let $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t)$ be the output of the game. We say that the trace and revoke scheme $\mathtt{TR} = (\mathcal{G}, \mathcal{Q}, \mathcal{F})$ is immune to pirate evolution with respect to key-leaking incident $\mathcal{L}$ if, for any probabilistic polynomial time adversary $\mathcal{P}$, any R and any $t \in \{1, \ldots, |\mathsf{N} - \mathsf{R}|\}$, it holds $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t) = t$. We define the pirate evolution bound $\mathsf{evo}[\mathtt{TR}]$ of a trace and revoke scheme $\mathtt{TR}$ as the supremum of all $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t)$, for any leaking incident $\mathcal{L}$, any set of revoked users R and any evolving pirate $\mathcal{P}$; note that $\mathsf{evo}[\mathtt{TR}]$ is a function*

$\langle \mathcal{I}_1, \mathcal{I}_2, \cdots \mathcal{I}_N \rangle \leftarrow \mathcal{G}(1^\lambda; \rho; N)$ where $\rho \leftarrow Coins$
$\mathsf{T} \leftarrow \mathcal{L}(\mathcal{I}_1, \mathcal{I}_2, \cdots \mathcal{I}_n, t, \mathsf{R})$; $\mathsf{K} = \{\mathcal{K}_u \mid u : \mathcal{I}_u \in \mathsf{T}\}$
$\mathcal{B} \leftarrow \mathcal{P}^{\mathcal{E}_\mathsf{R}(\mathcal{M})}(\mathsf{T}, \mathsf{K})$ where $\mathcal{E}_\mathsf{R}(\mathcal{M})$ is an oracle that returns $\mathcal{E}_\mathsf{R}(m)$ with $m \leftarrow \mathcal{M}$
$\ell = 0$
repeat $\ell = \ell + 1$
    $B_\ell \leftarrow \mathcal{B}(1^{t+\log N}, \ell)$
until $Prob[B_\ell(\mathcal{E}_\mathsf{R}^{B_1, B_2, \cdots B_{\ell-1}}(m)) = m] < q$ with $m \leftarrow \mathcal{M}$
output ℓ.

Fig. 1. The attack game played with an evolving pirate.

of t and possibly of other parameters as well. A scheme is susceptible *to pirate evolution if its pirate evolution bound satisfies* $\mathsf{evo}[\mathtt{TR}] > t$*.*

Note that immunity against pirate evolution attacks is possibly a stringent property; even though we show that it is attainable (cf. the next section) it could be sacrificed in favor of efficiency. Naturally, using a trace and revoke scheme that is susceptible to a pirate evolution that produces many pirate decoders may put the system's managers at a perilous condition once a leaking incident occurs (and as practice has shown leaking incidents are unavoidable). The decision to employ a particular trace-and-revoke scheme in a certain practical setting should be made based on a variety of requirements and constraints and the system's behavior with respect to pirate evolution should be one of the relevant parameters that must be considered in the security analysis.

3.1 A Trace and Revoke Scheme Immune to Pirate-Evolution

In this section we show a simple trace and revoke design that achieves immunity against pirate-evolution. The system simply encrypts the session key with the unique key of each user in the system that is not revoked. This kind of linear length trace and revoke scheme can be formalized in the context of the Subset-Cover framework as follows:

Definition 3. *Let $|\mathsf{S}_j| = 1$ for all $j \in \mathcal{J} = \{1, 2, \cdots N\}$, i.e. the collection is consist of single element sets. Thus, for any user u, $|\mathcal{K}_u| = |\{\mathsf{L}_u\}| = 1$ holds. The key assignment $\mathcal{G}(1^\lambda, N)$ is done by choosing a random key for each S_j. The encryption functions $\mathcal{Q}$ and $\mathcal{F}$ are encryption functions used in any Subset-Cover framework. We say such trace and revoke scheme $(\mathcal{G}, \mathcal{Q}, \mathcal{F})$ is called linear length scheme since the size of cover for non-revoked users in $\mathsf{N} \setminus \mathsf{R}$ will be linear in $|\mathsf{N}| - |\mathsf{R}|$.*

The header of a ciphertext $\mathcal{C}$ contains the encryption of session key $\mathcal{Q}_{\mathtt{L}_u}(K)$ if user $u \in \mathsf{N} \setminus \mathsf{R}$. Under the assumption of sufficiently strong $\mathcal{Q}(\cdot)$ no other user will be able recover session key through $\mathcal{Q}_{\mathtt{L}_u}(K)$ and a user u will not be able to recover session key unless the header contains $\mathcal{Q}_{\mathtt{L}_u}(K)$. We show that immunity to pirate evolution is achievable:

Theorem 1. *The trace and revoke scheme as defined in Definition 3 is immune to pirate evolution, i.e. for all polynomial-time adversaries $\mathcal{P}$ and for any key leaking incident $\mathcal{L}$, $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t) = t$.*

3.2 Pirate Evolution for the Complete Subtree Method

In this section, we demonstrate that the complete subtree method (CS) of [22] is susceptible to pirate evolution. Specifically, we present an evolving pirate that can produce up to $t \log N/t$ pirate boxes, given t traitor keys. Below we present some definitions that will be used throughout this section:

Definition 4. *The partition $\mathcal{S} = \mathcal{T}(\mathsf{R})$ is the set of subsets $\mathsf{S}_{i_1}, \mathsf{S}_{i_2}, \cdots \mathsf{S}_{i_m}$ where, $0 < j \leq m$, $i_j \in \mathcal{J}$ corresponds to a node in the full binary tree. Denote the root of subtree containing the users in $\mathsf{S} \in \{\mathsf{S}_j\}_{j \in \mathcal{J}}$ by $root(\mathsf{S})$, in general we will be using $root()$ as a function outputting root of a given tree. Suppose $\mathsf{T} = \mathcal{L}(\mathcal{I}_1, \cdots, \mathcal{I}_n, t, \mathsf{R})$, then we say the Steiner tree $ST(\mathsf{T}, \mathsf{S})$ is the minimal subtree of the binary tree rooted at $root(\mathsf{S})$ that connects all the leaves on which the users in $\mathsf{T} \cap \mathsf{S}$ are placed.*

We denote the unique key of a node v by $\mathtt{L}(v)$. It is possible for any $u \in \mathsf{S}$ to deduce $\mathtt{L}(root(\mathsf{S}))$ from its private information $\mathcal{I}_u, \mathcal{K}_u$. A pirate box $Box(\mathtt{L}(v))$ is a decoder that uses the key associated to S where $root(\mathsf{S}) = v$; it decrypts $\mathcal{E}_{\mathcal{S}}(\cdot)$ iff there exists a $\mathsf{S} \in \mathcal{S}$ s.t. $root(\mathsf{S}) = v$ holds. in other terms, $Box(\mathtt{L}(v))$ decrypts $\mathcal{C} = \mathcal{E}_{\mathsf{R}}(\cdot)$ iff the header of $\mathcal{C}$ contains the encryption $\mathcal{Q}_{\mathtt{L}(v)}(K)$.

Figure 2 is an illustration for the partition of non-revoked users and the set of traitors in a broadcasting scheme using the CS method. The description of the

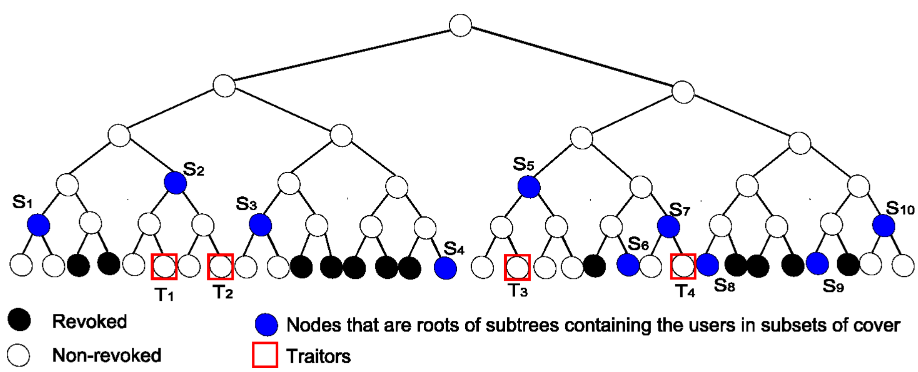

Fig. 2. Complete Subtree with cover and set of traitors

evolving pirate relies on a simple observation that is the following lemma:

Lemma 1. *For a given set of revoked users R let $\mathcal{T}^{P_i}(\mathsf{R})$ be the partition generated after tracing pirate box $P = Box(\mathtt{L}(v))$. Let S be the subset such that*

$root(\mathsf{S}) = v$ holds. Suppose $\mathsf{S} = \mathsf{S}_L \cup \mathsf{S}_R$ where subset S_L (resp. S_R) is left (resp. right) part of the subtree rooted at v. It holds that: $\mathsf{S} \in \mathcal{T}(\mathsf{R})$ if and only if $\mathsf{S}_L \in \mathcal{T}^P(\mathsf{R})$ and $\mathsf{S}_R \in \mathcal{T}^P(\mathsf{R})$.

According to the above lemma, the pirate will be able to produce a new version of pirate box after $P_i = Box(\mathsf{L}(root(\mathsf{S})))$ is caught. That is true because after tracing P_i, a traitor u is either in S_L or S_R, and the pirate still will be able to produce a new box by using the key associated to S_L(or S_R depends on which one contains u). The motivation of the evolving pirate is exploiting the above observation to successively generate pirate boxes.

We define the master pirate box $\mathcal{B}$ produced by the adversary $\mathcal{P}^{\mathcal{E}_\mathsf{R}(\mathcal{M})}(\mathsf{T},\mathsf{K})$ as producing a vector of pirate boxes. $\mathcal{B}$ constructs the sequence of pirate boxes by walking on the nodes of the forest of Steiner trees $\{ST(\mathsf{T},\mathsf{S}) \mid \mathsf{S} \in \mathcal{T}(\mathsf{R})\}$. More technically, it recursively runs a procedure called `makeboxes` on each $\mathsf{Tree} = ST(\mathsf{T},\mathsf{S})$ which first creates a pirate box `Box` by using the unique key assigned to the node $root(\mathsf{Tree})$. It then splits the Tree into two trees. The splitting is needed because tracing `Box` will result in the partition of the subset S. Thus the splitting procedure is based on the partition of subset S into two equal subsets (in `CS` tracing works by splitting into the two subtrees rooted at the children of $root(\mathsf{Tree})$). The master box $\mathcal{B}$ then runs `makeboxes` independently on both of the trees resulted from the partition. Figure 3 is the summary of the evolving pirate strategy. The number of generations that can be produced equals the number of nodes in the forest of Steiner trees $\{ST(\mathsf{T},\mathsf{S}) \mid \mathsf{S} \in \mathcal{T}(\mathsf{R})\}$.

1.	For each $\mathsf{S} \in \mathcal{T}(\mathsf{R})$ run `makeboxes`$(ST(\mathsf{T},\mathsf{S}))$ till the ℓ-th box is produced.
`makeboxes`(Tree)	
1.	Take any user u placed on a leaf of Tree.
2.	Output $Box(\mathsf{L}(root(\mathsf{Tree})))$ where $\mathsf{L}(root(\mathsf{Tree}))$ is available from $\mathcal{K}_u$
3.	Let ST_L and ST_R be respectively the left and right subtrees of Tree.
4.	run `makeboxes`(ST_L)
5.	run `makeboxes`(ST_R)

Fig. 3. The description of master box program $\mathcal{B}(1^{t+\log N}, \ell)$ parameterized by $\mathcal{T}(\mathsf{R}), \mathsf{T}, \mathcal{K}_u$ for $u \in \mathsf{T}$ that is produced by the evolving pirate for the complete subtree method.

Theorem 2 is formalizing and proving the correctness of the above procedure, i.e. the next generation should be able to decrypt the message encrypted after tracing has disabled all previous boxes.

Theorem 2. *Let $P_1, P_2, \cdots, P_{v+1}$ be a sequence of pirate boxes constructed by the evolving pirate strategy described in Figure 3. Suppose $\mathcal{C} = \mathcal{E}_\mathsf{R}^{P_1,P_2,\cdots,P_v}(M)$, then $Prob[P_{v+1}(\mathcal{C}) = M] \geq q$, provided that v is less than the number of nodes in the forest of trees $\{ST(\mathsf{T},\mathsf{S}) \mid \mathsf{S} \in \mathcal{T}(\mathsf{R})\}$.*

Leaking Incidents. For the polynomial time adversary $\mathcal{P}$ described in Figure 3, $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t)$ is the number of nodes in the forest of the Steiner trees of the traitors. Theorem 3 and Theorem 4 give some bounds on this quantity depending on the leaking incident.

Theorem 3. *Let* N *be the set of* N *users represented by a full binary tree in the Complete Subtree method. For a given* R*, any leaking incident* $\mathcal{L}$ *corrupting* t *users in a single subset* $\mathsf{S} \in \mathcal{T}(\mathsf{R})$ *enables an evolving pirate with respect to* $\mathcal{L}$ *so that* $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t) \geq 2t - 2 + \log(|\mathsf{S}|/t)$.

Theorem 4. *Let* N *be the set of* N *users represented by a full binary tree. For a given* R*, there exists a leaking incident* $\mathcal{L}$ *corrupting* t *users in a single subset* $\mathsf{S} \in \mathcal{T}(\mathsf{R})$ *so that* $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t) \geq 2t - 1 + t\log(|\mathsf{S}|/t)$.

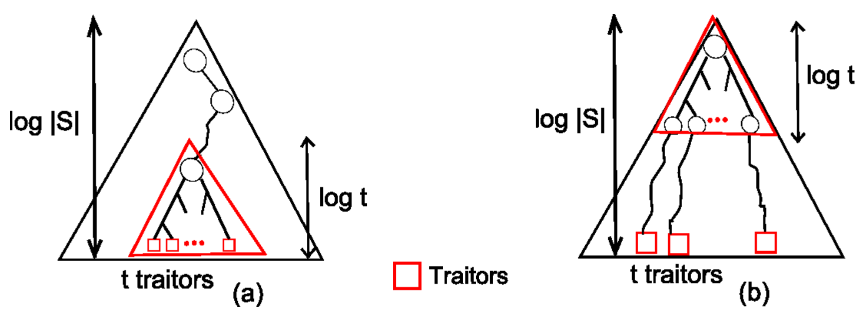

Fig. 4. The illustration of the bounds on the number of pirate boxes produced (a) All of the traitors are descendants of a same node with height $\log t$ (b) None of the paths with length $\log(|S|/t)$ from leaves to the root intersect.

Figure 4 shows the cases where it is possible to achieve the bounds given in above two theorems. Figure 4(a) is yielding the bound in Theorem 3 and, Figure 4(b) is yielding the bound in Theorem 4. The maximum number of generations can be achieved following Figure 4(b) in a configuration of the system when there is no revoked user; in this case there is a single element in the partition, namely S containing N users. It follows that the pirate can produce up to $t\log(N/t)$ generations and thus:

Corollary 1. *The pirate evolution bound for the* `CS` *method satisfies* $\mathsf{evo}[\mathtt{CS}] \geq t\log(N/t)$.

3.3 Pirate Evolution for the Subset Difference Method

In this section we turn our attention to the Subset Difference (`SD`) method of [22] that is part of the AACS standard [1]. Compared to the Complete Subtree method, the subsets in the `SD` method are represented by pairs of nodes. We define the required notations as follows:

Definition 5. *Let* $\mathsf{S}_{i,j} \in \{\mathsf{S}_{i,j}\}_{(i,j)\in\mathcal{J}}$ *be the set of all leaves in the subtree rooted at* v_i *but not of* v_j*. We define the function set* : $\{(v_i, v_j) \mid (i,j) \in \mathcal{J}\} \rightarrow \{\mathsf{S}_{i,j}\}_{(i,j)\in\mathcal{J}}$ *such that the inverse function* $set^{-1}()$ *maps a subset to its corresponding pair of nodes. Since* $\mathsf{S}_{i,j}$ *is somehow related to a tree, we still use* $root(\mathsf{S}_{i,j})$ *to output* v_i*. Suppose* $\mathsf{T} = \mathcal{L}(\mathcal{I}_1, \cdots, t, \mathsf{R})$*, then we say the Steiner tree* $ST(\mathsf{T}, v_i, v_j)$ *is the minimal subtree of the binary tree rooted at* v_i*, excluding the descendants of* v_j*, that connects all the leaves in* $\mathsf{T} \cap set(v_i, v_j)$ *and node* v_j*. A pirate box* $Box(v_i, v_j, \mathcal{K}_u)$ *is a decoder that uses the key associated to* $set(v_i, v_j)$ *as inferred by the private data* $\mathcal{K}_u$ *assigned to the user* u*. For simplicity, we also denote the pirate decoder by* $Box(v_i, v_j, u)$*(omitting* $\mathcal{K}_u$*). By definition;* $Box(v_i, v_j, u)$ *inverts* $\mathcal{E}_{\mathcal{S}}(\cdot)$ *iff there exists a* $\mathsf{S} \in \mathcal{S}$ *such that* $set(v_i, v_j) = \mathsf{S}$ *holds.*

The susceptibility of the `SD` method to pirate evolution relies on the following simple observation regarding the tracing algorithm and the way it operates on a given pirate box:

Lemma 2. *For a given set of revoked users* R *let* $\mathcal{T}^P(\mathsf{R})$ *be the partition generated after tracing pirate box* $P = Box(v_i, v_j, u)$*. Suppose* v_c *is the child of* v_i *that is on the path from* v_i *to* v_j*; note that in this case* $set(v_i, v_j) = set(v_i, v_c) \cup set(v_c, v_j)$*. It holds that:* $set(v_i, v_j) \in \mathcal{T}(\mathsf{R})$ *if and only if* $set(v_i, v_c) \in \mathcal{T}^P(\mathsf{R})$ *and* $set(v_c, v_j) \in \mathcal{T}^P(\mathsf{R})$*.*

We will exploit the above lemma to successively generate pirate boxes. This is possible because after tracing $P = Box(v_i, v_j, u)$, the traitor u is still in one of the subsets in the partition $\mathcal{T}^P(\mathsf{R})$. We will present an evolving pirate strategy based on the forest of Steiner trees $\{ST(\mathsf{T}, v_i, v_j) \mid set(v_i, v_j) \in \mathcal{T}(\mathsf{R})\}$ by walking on the paths of Steiner trees that will be predefined according to a scheduling of traitors. Unlike our evolving pirate strategy for the `CS` method, we are focusing on paths instead of nodes because of the inherent structure of the `SD` method and the way tracing works by merging subsets under a simple condition that is shown in lemma 3. The merging performed by NNL (whose main objective is to curb the ciphertext expansion) is, as we observe, an opportunity for pirate evolution as it leads to the reuse of some nodes in different pairs, i.e. different subsets in the collection $\{\mathsf{S}_{i,j}\}_{(i,j)\in\mathcal{J}}$. In general, the merging will occur whenever it is allowed based on lemma 3 with the following exception: S_1 will not be merged if the partition contains another subset S_2 such that they have resulted from a split-up of a single subset at an earlier iteration of the subset-tracing procedure. The reader may refer to [22].

Lemma 3. *Let* $V_i, v_1, v_2, \cdots v_d, V_j$ *be any sequence of vertices which occur in this order along some root-to-leaf path in the tree corresponding to the subset* $set(V_i, V_j)$*, then* $set(V_i, V_j) = set(V_i, v_1) \cup set(v_1, v_2) \cup \cdots \cup set(v_d, V_j)$*.*

According to the way tracing works on the `SD`, whenever the partition contains a series of subsets $\{set(v_i, v_1), set(v_1, v_2), \cdots, set(v_d, v_j)\}$ they can potentially be merged using Lemma 3 into one single subset $set(v_i, v_j)$. To illustrate how evolution for SD method works, we first give an example of a partition of non-revoked users and the set of traitors in Figure 5. Let's focus on the subset rooted at g that is magnified in Figure 6(a) and start creating pirate boxes.

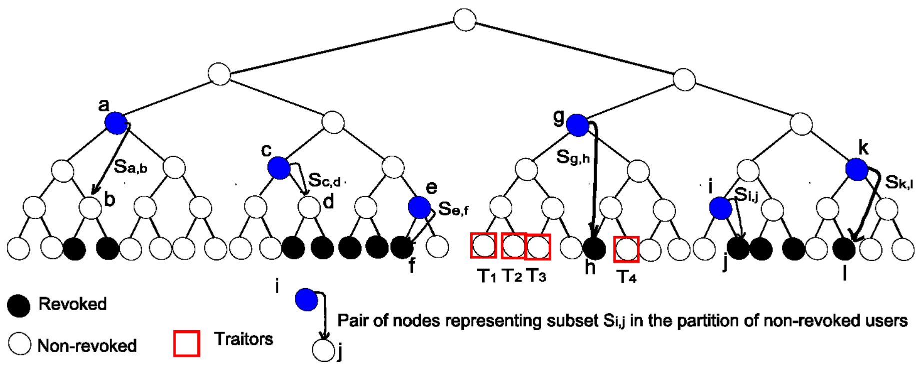

Fig. 5. Subset Difference with cover and set of traitors

Suppose that the evolving pirate uses the keys of the traitor T_4 first; the sequence of pirate boxes created until T_4 is entirely revoked would be $B_1 = Box(1,5,T_4), B_2 = Box(2,5,T_4)$ and $B_3 = Box(3,5,T_4)$. Due to lemma 2 tracing all these boxes would end up with revoking T_4 and $\mathcal{T}^{B_1,B_2,B_3}(\mathsf{R}) = \{set(1,2), set(2,3)\}$. Note that in light of lemma 3 the tracing algorithm will merge these two subsets to have the single subset $set(1,3)$ shown in Figure 6(b). This illustrates the fact that an evolving pirate against the SD method may use the keys of a traitor as many times as the height of the subset it belongs to *without* necessarily restricting the same opportunity for other traitors that are scheduled to be used later. Indeed, we can execute a pirate box construction using the keys of traitor T_3 that would be as many as the height of the tree (compare this to the Complete Subtree method where this is not achievable and using the keys of one traitors strips the opportunity to use such keys for other traitors scheduled later). Proceeding with our example, the master pirate box $\mathcal{B}$ will now be able to create a pirate box $Box(1,3,T_3)$ (recall that $\mathcal{T}^{B_1,B_2,B_3}(\mathsf{R}) = \{set(1,3)\}$) followed by another box $Box(1,2,T_3)$ and so on until T_3 is entirely revoked. Even though we have the opportunity now to make more boxes per traitor compared to the complete subset method, special care is needed to choose the order with which we are expending the traitor keys as we will illustrate below. This is in sharp contrast to the complete subset method where the scheduling of traitors makes no difference in terms of the number of pirate box generations that the master box can spawn. To see the importance of scheduling the traitors appropriately, suppose that we use the traitor T_3 first instead of T_4; then, the sequence of pirate boxes created until T_3 is entirely revoked would be $B_1 = Box(1,5,T_3), B_2 = Box(1,2,T_3), B_3 = Box(8,9,T_3)$ and $B_4 = Box(10,12,T_3)$ (refer to figure 6(b) for the node numbering). Tracing all these boxes would end up with revoking T_3

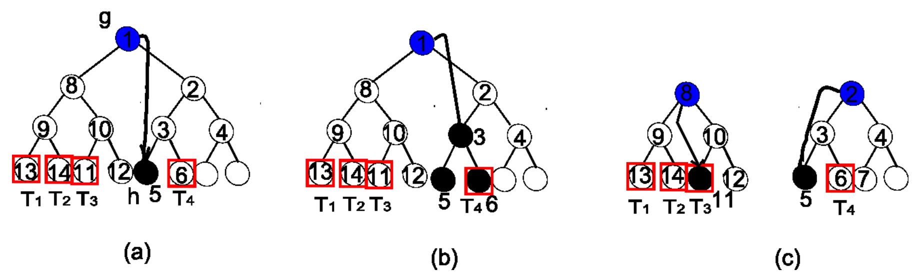

Fig. 6. (a) The subset $\mathsf{S}_{1,5}$ of Figure 5 that contains all the traitors. The master pirate box will start producing boxes according to this subset. (b) The partition after pirate evolution using T_4 took place; T_4 is revoked from the system. (c) The partition after pirate evolution using T_3 took place; T_3 is revoked from the system.

and $\mathcal{T}^{B_1,B_2,B_3,B_4}(\mathsf{R}) = \{set(2,5), set(8,10), set(10,11)\}$. Note that this subset collection will also be merged by tracing algorithm, resulting the partition given in figure 6(c). The pirate then will be able to create a pirate box $Box(2,5,T_4)$ and so on until T_4 is revoked. Observe that now T_4 is isolated in its own subtree, and the master pirate box will be able to make fewer boxes using the keys of T_4. Thus, it would have been preferable to start the pirate evolution with traitor T_4.

We observe that the evolving pirate strategy can be based on a representation of the Steiner tree by means of paths hanging off each other hierarchically such that each path stems from an internal node to a traitor placed on a leaf. Each time we choose a traitor, we actually choose a path to walk on to construct pirate boxes. We observe two criteria to maximize the number of pirate decoders. (1) Once a traitor is revoked, we choose a shortest path hanging off the path containing the recently revoked traitor. (2) If there are more than one shortest path, a path with large number of paths hanging off itself would be preferable. Choosing a traitor amounts to choosing a path according to this criteria (in a recursive way). In the next paragraphs we formalize these observations. We introduce a special annotation of the Steiner Tree $ST(\mathsf{T}, u, v)$, where $set(u, v)$ is one of the subsets in the partition, that will enable us to choose the best ordering of the traitors.

Definition 6. *A traitor annotation of a Steiner tree $ST(\mathsf{T}, u, v)$ is the mapping from its nodes to $\mathsf{T} \cup \{\perp\}$ that is defined in Figure 7. We say $ST(\mathsf{T}, u, v)$ is annotated by f. Denote the parent of a node v by $parent(v)$, the sibling by $sibling(v)$, the height by $height(v)$. We define the rank of a traitor s given an annotation f as the number of nodes with 2 children that are annotated by s. We denote the rank of $s \in \mathsf{T}$ by $rank(s)$. Given a Steiner tree ST annotated by f, for any $u \in \mathsf{T}$ the u-$\mathsf{path}(\mathsf{ST})$ is the path that is made out of all nodes that are annotated by u. Similarly, we define $\perp$-$\mathsf{path}(\mathsf{ST})$ and further we call it as the basic path of the tree ST. We denote u-$\mathsf{path}(\mathsf{ST})$ by a vector of nodes, u-$\mathsf{path}(\mathsf{ST}) = \langle v_1, v_2, \cdots v_s \rangle$ where $v_i = parent(v_{i+1})$ for $0 < i < s$ and $u = f(v_i)$; we also denote v_1 and v_s in this path by $top_f(u)$ and $bottom_f(u)$ respectively.*

$\texttt{annotation}(\mathsf{Tree}\ ST(\mathsf{T}, i, j))$
Initially annotate each leaf l with its corresponding traitor $u \in \mathsf{T}$, i.e. $f(l) = u$
$rank(u) = 0$, for each $u \in \mathsf{T}$
$f(j) = \perp$ and $rank(\perp) = 0$.
Annotate each node from bottom to top by following rule:
$f(parent(v)) = \left\{ \begin{array}{ll} f(v) & sibling(v) \notin \mathsf{Tree} \vee f(v) = \perp \\ f(v) & rank(f(v)) \geq rank(f(sibling(v))) \\ f(sibling(v)) & \text{otherwise} \end{array} \right\}$
update $rank(f(parent(v))) = rank(f(parent(v))) + 1$ if $sibling(v) \in \mathsf{Tree}$
output f

Fig. 7. Computing the traitor annotation for a given Steiner tree

Lemma 4. *For a given set of revoked users* R *and the set of traitors* T*, let* $\mathsf{Tree} = ST(\mathsf{T}, v_i, v_j) \in \{ST(\mathsf{T}, g, r) \mid set(g, r) \in \mathcal{T}(\mathsf{R})\}$ *be one of the Steiner trees. Consider the annotation of* Tree *given in Figure 7. Suppose the shortest path hanging off the* $\perp$-$\mathsf{path}(\mathsf{Tree})$ *is annotated by* u*. Let* $u_1 = top_f(u)$ *and* $u_s = bottom_f(u)$ *where* s *is the length of the* u*-path(*Tree*). It holds that: (1) There exists a sequence of pirate boxes* $B_1, B_2, \cdots B_k$*, each using a private key derived from* $\mathcal{K}_u$ *where* $k = height(\mathsf{Tree}) + A_u$ *and* $A_u \in \{0, 1\}$ *such that* $A_u = 1$ *if and only if* $sibling(u_1)$ *has a single child in* Tree*. (2)* $\mathcal{T}^{B_1, B_2, \cdots B_k}(\mathsf{R}) = \{\mathcal{T}(\mathsf{R}) \setminus set(v_i, v_j)\} \cup \{set(v_i, parent(u_1)), set(u_1, u_s)\}$*.*

We next describe our evolving pirate strategy against the SD method. We define the master pirate box $\mathcal{B}$ produced by the adversary $\mathcal{P}^{\mathcal{E}_\mathsf{R}(\mathcal{M})}(\mathsf{T}, \mathsf{K})$ as follows: $\mathcal{B}$ recursively runs a procedure for each subset $\mathsf{S} = set(v_i, v_j) \in \mathcal{T}(\mathsf{R})$ which is called `makeboxes`, with input the traitor annotated Steiner tree $\mathsf{Tree} = ST(\mathsf{T}, v_i, v_j)$. Observe below that whenever the recursive call is made, the annotation of Tree satisfies that the root is annotated with $\perp$. The basic procedure works as follows:

The root v_i is annotated as $\perp$. Let u-$\mathsf{path}(\mathsf{Tree})$=$\langle u_1, u_2, \cdots u_s\rangle$ be the shortest path hanging off the $\perp$-$\mathsf{path}(\mathsf{Tree})$. The master box $\mathcal{B}$ constructs $Box(v_i, v_j, u)$ and more pirate decoders by applying lemma 2. After creating pirate boxes as many as the height of Tree (plus one possibly if $A_u = 1$, cf. lemma 4), the traitor u will be entirely revoked by the system. Lemma 4 tells us that the partition after revoking u will include the subsets $set(v_i, parent(u_1))$ and $set(u_1, u_s)$. We update the path $\langle u_1, u_2, \cdots u_s\rangle$ in $ST(\mathsf{T}, u_1, u_s)$ by annotating it $\perp$ since u is no more in $(set(u_1, u_s))$. The master box $\mathcal{B}$ then runs `makeboxes` independently on both of the trees $ST(\mathsf{T}, v_i, parent(u_1))$ and $ST(\mathsf{T}, u_1, u_s)$. Refer to figure 8 for the detailed specification of the evolving pirate strategy.

In the following theorem we prove the correctness of the strategy, i.e. that each box will decrypt the ciphertexts that are generated assuming all previous boxes are traced. We also show the maximum number of pirate decoders that can be created.

Theorem 5. *Let* $P_1, P_2, \cdots, P_{k+1}$ *be a sequence of pirate boxes constructed by the pirate evolution strategy described in Figure 8. Suppose* $\mathcal{C} = \mathcal{E}_\mathsf{R}^{P_1, P_2, \cdots, P_k}(M)$*,*

1. For each $\mathsf{S}_{i,j} = set(v_i, v_j) \in \mathcal{T}(\mathsf{R})$
2. Compute $f = \mathtt{annotation}(ST(\mathsf{T}, v_i, v_j))$
3. Run $\mathtt{makeboxes}(ST(\mathsf{T}, v_i, v_j), f)$ till the ℓ-th pirate box is produced.

$\mathtt{makeboxes}(\mathsf{Tree}$, annotation $f)$

1. Let $\perp$-path in Tree be $\langle k_1, k_2, \cdots k_m\rangle$. Note that $k_1 = v_i$ and $k_m = v_j$
2. Choose the shortest path hanging off the $\perp$-path, i.e. pick $l = \max(l : sibling(k_l) \in \mathsf{Tree})$ to use the keys of traitor $u = f(sibling(k_l))$; if no such path exists, exit.
3. Denote u-path by $\langle u_1, u_2, \cdots u_s\rangle$
4. Output $Box(k_1, k_m, u), Box(k_2, k_m, u), \cdots Box(k_{l-1}, k_m, u)$
5. Output $Box(k_{l-1}, k_l, u)$ iff $l < m$.
6. Output $Box(u_1, sibling(u_2), u), Box(u_2, sibling(u_3), u), ..Box(u_{s-1}, sibling(u_s), u)$
8. Update $f(u_i) = \perp$, for $0 < i \leq s$
9. $\mathtt{makeboxes}(ST(\mathsf{T}, u_1, u_s), f)$
10. $\mathtt{makeboxes}(ST(\mathsf{T}, k_1, k_{l-1}), f)$

Fig. 8. The description of master box program $\mathcal{B}(1^{t+\log N}, \ell)$ parameterized by $\mathcal{T}(\mathsf{R})$, T, $\mathcal{K}_u$ for $u \in \mathsf{T}$ that is produced by the evolving pirate for the Subset Difference method.

then $Prob[P_{k+1}(\mathcal{C}) = M] \geq q$, *provided that*

$$k < \sum_{set(v_i,v_j)\in\mathcal{T}(\mathsf{R})} \left(rank(\perp)\cdot height(v_i) + \sum_{u\in\mathsf{T}\cap set(v_i,v_j)} C_u + A_u \right)$$

where $C_u = rank(u)\cdot \mid u\text{-}\mathsf{path}(ST(T, v_i, v_j)) \mid$.

Leaking Incidents. For the evolving pirate $\mathcal{P}$ described in Figure 8, the value of $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t)$ follows from theorem 5. Theorem 6 gives some bounds on this quantity depending on the leaking incident for the $\mathtt{SD}$ method.

Theorem 6. *Let* N *be the set of* N *users represented by a full binary tree in the* $\mathtt{SD}$ *method. For a given* R*, there exists a leaking incident* $\mathcal{L}$ *corrupting* t *users in* $\mathsf{S} \in \mathcal{T}(\mathsf{R})$*(for simplicity assume* S *is complete subset, and thus* $\log|\mathsf{S}|$ *is an integer), that enables an evolving pirate with respect to* $\mathcal{L}$ *so that*

$$PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t) = \begin{cases} t\log|\mathsf{S}|, & t \leq \log|\mathsf{S}| + 1 \\ t\log(\frac{|\mathsf{S}|}{2^m}) + 2^m\log(\frac{|\mathsf{S}|}{2^{m-3}}) - \log|\mathsf{S}| - 3, & \end{cases}$$

$$t \in \left\{2^{m-1}\log(\frac{|\mathsf{S}|}{2^{m-2}}){+}1, \ldots, 2^m\log(\frac{|\mathsf{S}|}{2^{m-1}})\right\} \text{ for } 0 < m < \log|\mathsf{S}|$$

To see the above existence result, consider the following: our goal is to choose t traitors in the set S. Once a leaf is chosen in a complete subtree to place a traitor, we will next choose to place other traitors in the subtrees hanging off the path of that traitor.The first traitor chosen in each hanging subtree, it contributes to the number $PE^{\mathsf{R}}_{\mathcal{P},\mathcal{L}}(t)$ as many pirate generations as the height of the tree (say $h = \log|\mathsf{S}|$). After this first stage of placement, we have h subtrees each containing one traitor and each have different heights. We recursively place the remaining traitors in these subtrees by using the highest possible subtrees first. In short, we can think of the leaking incident of theorem 6 as follows: the first traitor is placed in an arbitrary leaf; then, h stages follow: in stage m (where $m = 0, \ldots, h-1$), the remaining traitors are placed on the subtrees of height $h - m$. At stage $m > 0$ we can place $2^{m-1}(h-m)$ traitors (where we place h traitors at stage 0). By the end of stage m we will have already placed $2^m \log(\frac{|\mathsf{S}|}{2^{m-1}})$ traitors. Note that a traitor placed at stage m will contribute $h-m$ pirate boxes following our evolving pirate strategy. The formula in theorem 6 gives the sum of all pirate generations for each traitor.

The maximum number of generations can be achieved following the leaking incident of Theorem 6 in a configuration of the system when there is no revoked user; in this case there is a single element in the partition, namely S containing N users. The corollary below follows easily from theorem 6.

Corollary 2. *The pirate evolution bound for the* SD *method satisfies* $\mathsf{evo}[\mathsf{SD}] \geq t \log N$ *for* $t \leq \log N$*. It also satisfies that* $\mathsf{evo}[\mathsf{SD}] \geq t \frac{\log N}{2}$ *for* $t \leq \sqrt{N} \cdot \frac{\log N}{2}$.

Relation to the AACS. The AACS standard for Blue-Ray disks and HD-DVDs uses the SD method with $N = 2^{31}$ nodes. It follows that a leaking incident with t traitors enables our evolving pirate strategy to generate up to $31 \cdot t$ generations of pirate boxes in the case that the system has an initial state of ciphertexts with a single element in the partition; note that if the starting configuration of the system has more elements in the partition (e.g., 2^8 elements each corresponding to 2^{23} users) the total number of generations would be $23 \cdot t$ for $t \leq 23$, and so on.

References

1. AACS Specifications, 2006 `http://www.aacsla.com/ specifications/`.
2. D. Boneh and M. Franklin, An Efficient Public-Key Traitor Tracing Scheme, CRYPTO '99, LNCS 1666 Springer 1999. pp. 338-353.
3. D. Boneh, A. Sahai and B. Waters, Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. EUROCRYPT 2006,LNCS 4004, pp. 573-592.
4. D. Boneh and J. Shaw, Collusion-Secure Fingerprinting for Digital Data, IEEE Transactions on Information Theory, Vol. 44(5) pp. 1897-1905, 1998.
5. H. Chabanne, D. Hieu Phan and D. Pointcheval, Public Traceability in Traitor Tracing Schemes, EUROCRYPT 2005, LNCS 3494 Springer 2005, pp. 542-558.
6. B. Chor, A. Fiat, and M. Naor, Tracing Traitors, CRYPTO '94, LNCS 839 Springer 1994, pp. 257-270.
7. B. Chor, A. Fiat, M. Naor, and B. Pinkas, Tracing Traitors, IEEE Transactions on Information Theory, Vol. 46, 3, 893-910, 2000.
8. Y. Dodis, N. Fazio: Public Key Broadcast Encryption for Stateless Receivers.Security and Privacy in Digital Rights Management,DRM 2002, Revised Papers, pp. 61-80.
9. Y. Dodis, N. Fazio, A. Kiayias, M. Yung: Scalable public-key tracing and revoking, PODC 2003, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), July 13-16, 2003, Boston, Massachusetts.
10. A. Fiat, M. Naor: Broadcast Encryption. CRYPTO '93, LNCS 773 Springer 1994.

11. A. Fiat and T. Tassa, Dynamic Traitor Tracing. Journal of Cryptology Vol. 4(3), pp. 211-223, 2001.
12. O. Goldreich, S. Goldwasser and S. Micali, How to Construct Random Functions, J. of the ACM 33(4), 1986, pp. 792-807.
13. E. Gafni, J. Staddon and Y. Lisa Yin, Efficient Methods for Integrating Traceability and Broadcast Encryption, CRYPTO '99,LNCS 1666 Springer 1999, pp. 372-387.
14. J. A. Garay, J. Staddon, A. Wool: Long-Lived Broadcast Encryption. CRYPTO 2000,LNCS 1880 Springer 2000, pp 333-352.
15. D. Halevy and A. Shamir, The LSD Broadcast Encryption Scheme. CRYPTO 2002, LNCS 2442 Springer 2002, pp. 47-60.
16. N. Jho, J. Y. Hwang, J. Hee Cheon, M. Hwan Kim, D. Hoon Lee, E. Sun Yoo, One-Way Chain Based Broadcast Encryption Schemes.EUROCRYPT 2005,LNCS 3494 Springer 2005, pp. 559-574.
17. A. Kiayias, M. Yung: Self Protecting Pirates and Black-Box Traitor Tracing.CRYPTO 2001,LNCS 2139 Springer 2001, pp. 63-79.
18. A. Kiayias and M. Yung, On Crafty Pirates and Foxy Tracers, ACM CCS-8 Workshop DRM 2001, LNCS 2320 Springer 2002, pp. 22-39.
19. A. Kiayias, M. Yung, Traitor Tracing with Constant Transmission Rate,EUROCRYPT 2002,LNCS 2332 Springer 2002, pp. 450-465.
20. K. Kurosawa and Y. Desmedt, Optimum Traitor Tracing and Asymmetric Schemes, EUROCRYPT '98 LNCS 1403, Springer 1998, pp. 145-157.
21. D. Micciancio and S. Panjwani, Corrupting One vs. Corrupting Many: The Case of Broadcast and Multicast Encryption. ICALP 2006,LNCS 4052 Springer 2006.
22. D. Naor, M. Naor, and J. B. Lotspiech Revocation and Tracing Schemes for Stateless Receivers, CRYPTO 2001, LNCS 2139 Springer 2001, pp. 41–62.
23. M. Naor and B. Pinkas, Threshold Traitor Tracing, CRYPTO '98,LNCS 1462 Springer 1998, pp. 502-517.
24. M. Naor and B. Pinkas, Efficient Trace and Revoke Schemes, FC 2000 LNCS 1962 Springer 2001, pp. 1–20.
25. M. Naor and O. Reingold, Number-Theoretic Constructions of Efficient Pseudo-Random Functions,FOCS '97, IEEE Computer Society, pp. 458-467.
26. B. Pfitzmann, Trials of Traced Traitors,Information Hiding,LNCS 1174 Springer 1996, pp. 49–63.
27. D. Hieu Phan, R. Safavi-Naini, D. Tonien: Generic Construction of Hybrid Public Key Traitor Tracing with Full- Public-Traceability. 264-275
28. R. Safavi-Naini and Y. Wang, Sequential Traitor Tracing, CRYPTO 2000 LNCS1880 Springer 2000. pp. 316-332.
29. R. Safavi-Naini and Y. Wang, Collusion Secure q-ary Fingerprinting for Perceptual Content,DRM 2001 LNCS 2320 Springer 2002, pp. 57–75.
30. R. Safavi-Naini and Y. Wang, New Results on Frameproof Codes and Traceability Schemes, IEEE Transactions on Information Theory, Vol. 47(7), pp. 3029-3033, 2001.
31. R. Safavi-Naini and Y. Wang, Traitor Tracing for Shortened and Corrupted Fingerprints. ACM CCS-9 Workshop, DRM 2002 LNCS 2696 Springer 2003, pp. 81-100.
32. J. N. Staddon, D. R. Stinson and R. Wei, Combinatorial Properties of Frameproof and Traceability Codes, IEEE Transactions on Information Theory, Vol. 47(3), pp. 1042-1049, 2001.
33. D. R. Stinson and R. Wei, Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes, SIAM Journal on Discrete Math, Vol. 11(1), pp. 41–53, 1998.
34. G. Tardos, Optimal probabilistic fingerprint codes,ACM 2003, pp. 116-125.