Revocation and Tracing Schemes for Stateless Receivers [⋆]

Dalit Naor[1], Moni Naor[2⋆⋆], and Jeff Lotspiech[1]

[1] IBM Almaden Research Center
650 Harry Road, San-Jose, CA. 95120
{lots, dalit}@almaden.ibm.com
[2] Department of Computer Science and Applied Math
Weizmann Institute, Rehovot Israel.
naor@wisdom.weizmann.ac.il

Abstract. We deal with the problem of a center sending a message to a group of users such that some subset of the users is considered revoked and should not be able to obtain the content of the message. We concentrate on the *stateless receiver* case, where the users do not (necessarily) update their state from session to session. We present a framework called the *Subset-Cover* framework, which abstracts a variety of revocation schemes including some previously known ones. We provide sufficient conditions that guarantees the security of a revocation algorithm in this class.

We describe two explicit Subset-Cover revocation algorithms; these algorithms are very flexible and work for any number of revoked users. The schemes require storage at the receiver of $\log N$ and $\frac{1}{2}\log^2 N$ keys respectively (N is the total number of users), and in order to revoke r users the required message lengths are of $r \log N$ and $2r$ keys respectively. We also provide a general *traitor tracing* mechanism that can be integrated with any Subset-Cover revocation scheme that satisfies a "bifurcation property". This mechanism does not need an a priori bound on the number of traitors and does not expand the message length by much compared to the revocation of the same set of traitors.

The main improvements of these methods over previously suggested methods, when adopted to the stateless scenario, are: (1) reducing the message length to $O(r)$ *regardless* of the coalition size while maintaining a single decryption at the user's end (2) provide a seamless integration between the revocation and tracing so that the tracing mechanisms does not require any change to the revocation algorithm.

Keywords: Broadcast Encryption, Revocation scheme, Tracing scheme, Copyright Protection.

⋆ A full version of the paper is available at the IACR Crypto Archive http://eprint.iacr.org/ and at http://www.wisdom.weizmann.ac.il/~naor/

⋆⋆ Work done while the author was visiting IBM Almaden Research Center and Stanford University. Partially supported by DARPA contract F30602-99-1-0530.

1 Introduction

The problem of a Center transmitting data to a large group of receivers so that only a predefined subset is able to decrypt the data is at the heart of a growing number of applications. Among them are pay-TV applications, multicast communication, secure distribution of copyright-protected material (e.g. music) and audio streaming. The area of Broadcast Encryption deals with methods to efficiently broadcast information to a dynamically changing group of users who are allowed to receive the data. It is often convenient to think of it as a Revocation Scheme, which addresses the case where some subset of the users are excluded from receiving the information. In such scenarios it is also desirable to have a Tracing Mechanism, which enables the efficient tracing of leakage, specifically, the source of keys used by illegal devices, such as pirate decoders or clones.

One special case is when the receivers are stateless. In such a scenario, a (legitimate) receiver is not capable of recording the past history of transmissions and change its state accordingly. Instead, its operation must be based on the current transmission and its initial configuration. Stateless receivers are important for the case where the receiver is a device that is not constantly on-line, such as a media player (e.g. a CD or DVD player where the "transmission" is the current disc), a satellite receiver (GPS) and perhaps in multicast applications. The stateless scenario is particularly relevant to the application of Copyright Protection.

This paper introduces very efficient revocation schemes which are especially suitable for stateless receivers. Our approach is quite general. We define a framework of such algorithms, called Subset-Cover algorithms, and provide a sufficient condition for an algorithm in this family to be secure. We suggest two particular constructions of schemes in this family; the performance of the second method is substantially better than any previously known algorithm for this problem (see Section 1.1). We also provide a general property ('bifurcation') of revocation algorithms in our framework that allows efficient tracing methods, *without* modifying the underlying revocation scheme.

Notation: Let N be the total number of users in the system let r be the size of the revoked set $\mathcal{R}$.

1.1 Related Work

Broadcast Encryption. The area of Broadcast Encryption was first formally studied (and coined) by Fiat and Naor in [12] and has received much attention since then. To the best of our knowledge the scenario of stateless receivers has not been considered explicitly in the past in a scientific paper. In principle any scheme that works for the connected mode, where receivers can remember past communication, may be converted to a scheme for stateless receivers (such a conversion may require to include with any transmission the entire 'history' of revocation events). Hence, when discussing previously proposed schemes we will consider their performance as adapted to the stateless receiver scenario.

A parameter that was often considered is t, the upper bound on the size of the coalition an adversary can assemble. The algorithms in this paper do not require such a bound and we can think of $t = r$; on the other hand some previously proposed schemes depend on t but are independent of r. The Broadcast Encryption method of [12] allows the removal of any number of users as long as at most t of them collude; the message length is $O(t \log^2 t)$, a user must store a number of keys that is logarithmic in t and is required to perform $\tilde{O}(r/t)$ decryptions.

The logical-tree-hierarchy (LKH) scheme, suggested independently by Wallner et al. [29] and Wong et al. [30], is designed for the connected mode for multicast applications. If used in the stateless scenario it requires to transmit $2r \log N$, store $\log N$ keys at each user and perform $r \log N$ encryptions (these bounds are somewhat improved in [5, 6, 20]). The key assignment of this scheme and the key assignment of our first method are similar (see Sect. 3.1 for comparison).

Luby and Staddon [19] considered the information theoretic setting and devised bounds for any revocation algorithms under this setting. Their "Or Protocol" fits our Subset-Cover framework; our second algorithm (the Subset Difference method) which is *not* information theoretic, beats their lower bound (Theorem 12 in [19]). In Garay et al. [16] keys of compromised decoders are no longer used and the scheme is adapted so as to maintain security for the good users. The method of Kumar et. al. [18] enables one-time revocation of up to r users with message lengths of $O(r \log N)$ and $O(r^2)$. CPRM [10] is one of the methods that explicitly considers the stateless scenario.

Tracing Mechanisms. Tracing systems, introduced by Chor et al. [8] and later refined to the Threshold Traitor model [23], [9], distribute decryption keys to the users so as to allow the detection of at least one 'identity' of a key that is used in a pirate box which was constructed using keys of at most t users. *Black-box tracing* assumes that only the outcome of the decoding box can be examined. The construction of [23] guarantees tracing with high probability; it required $O(t \log N)$ keys at each user, a single decryption operation and message length is $4t$. The public key tracing scheme of Boneh and Franklin [3] provides a number-theoretic deterministic method for tracing. Note that in all of the above methods t is an *a-priori* bound. Another notion, the one of *Content Tracing*, attempts to detect illegal users who redistribute the content *after* it is decoded (see [4, 13, 2, 26]).

Integration of tracing and revocation. Broadcast encryption can be combined with tracing schemes to yield trace-and-revoke schemes[1], a powerful approach to prevent illegal leakage of keys (others include the *legal* approach [25] and the *self enforcement* approach [11]). While Gafni et al. [15] and Stinson and Wei [28] consider combinatorial constructions, the schemes in Naor and Pinkas [24] are computational constructions and hence more general. The previously best known

[1] However it is *not* the case that every system which enables revocation and enables tracing is a trace-and-revoke scheme.

trace-and-revoke algorithm of [24] can tolerate a coalition of at most t users. It requires to store $O(t)$ keys at each user and to perform $O(r)$ decryptions; the message length is r keys, however these keys are elements in a group where the Decisional Diffie-Hellman problem is difficult, and hence these keys are longer than symmetric ones. The tracing model of [24] is not a "pure" black-box model. (Anzai et al. [1] employs a similar method for revocation, but without tracing capabilities.)

1.2 Summary of Results

In this paper we define a generic framework encapsulating several previously proposed revocation methods (e.g. the "Or Protocol" of [19]), called Subset-Cover algorithms. These algorithms are based on the principle of covering all non-revoked users by disjoint subsets from a predefined collection, together with a method for assigning (long-lived) keys to subsets in the collection. We define the security of a revocation scheme and provide a sufficient condition (key-indistinguishability) for a revocation algorithm in the Subset-Cover Framework to be secure. An important consequence of this framework is the separation between long-lived keys and short-term keys. The framework can be easily extended to the public-key scenario.

We provide two new instantiations of revocation schemes in the Subset-Cover Framework, with a different performance tradeoff (summarized in Table 1.2[2]). Both instantiations are tree-based, namely the subsets are derived from a virtual tree structure imposed on all devices in the system[3]. The first requires a message length of $r \log N$ and storage of $\log N$ keys at the receiver and constitutes a moderate improvement over previously proposed schemes; the second exhibits a substantial improvement: it requires a message length of $2r - 1$ (in the worst case, or $1.38r$ in the average case) and storage of $\frac{1}{2}\log^2 N$ keys at the receiver. This improvement is (provably) due to the fact that the key assignment is computational and not information theoretic (for the information theoretic case there exists a lower bound which exhibits its limits, see [21]). Furthermore, these algorithms are r-flexible, namely they do not assume an upper bound of the number of revoked receivers.

Thirdly, we present a tracing mechanism that works in tandem with a Subset-Cover revocation scheme. We identify the *bifurcation property* for a Subset-Cover scheme. Our two constructions of revocation schemes posses this property. We show that every scheme that satisfies the bifurcation property can be combined with the tracing mechanism to yield a trace-and-revoke scheme. The integration

[2] Note that the comparison in the processing time between the two methods treats an application of a pseudo-random generator and a lookup operation as having the same cost, even though they might be quite different. More explicitly, the processing of both methods consists of $O(\log \log N)$ lookups; in addition, the Subset Difference method requires at most $\log N$ applications of a pseudo-random generator.

[3] An alternative view is to map the receivers to points on a line and the subsets as segments.

of the two mechanisms is seamless in the sense that no change is required for any one of them. Moreover, no a-priori bound on the number of traitors is needed for our tracing scheme. In order to trace t illegal users, the first revocation method requires a message length of $t \log N$, and the second revocation method requires a message length of $5t$.

Main Contributions: the main improvements that our methods achieve over previously suggested methods, when adopted to the stateless scenario, are:

- Reducing the message length to linear in r regardless of the coalition size, while maintaining a single decryption at the user's end. This applies also to the case where public keys are used, *without* a substantial length increase.
- The seamless integration between revocation and tracing: the tracing mechanism does not require any change of the revocation algorithm and no a priori bound on the number of traitors, even when all traitors cooperate among themselves.
- The rigorous treatment of the security of such schemes, identifying the effect of parameter choice on the overall security of the scheme.

Method	Message Length	Storage@Receiver	Processing time	decryptions
Complete Subtree	$r \log \frac{N}{r}$	$\log N$	$O(\log \log N)$	1
Subset Difference	$2r - 1$	$\frac{1}{2} \log^2 N$	$O(\log N)$	1

Fig. 1. Performance tradeoff for the Complete Subtree method and the Subset Difference method

Organization of the paper. Section 2 describes the framework for Subset-Cover algorithms and a sketch of the main theorem characterizing the security of a revocation algorithm in this family (the security is described in details in the full version of the paper). Section 3 describes two specific implementations of such algorithms. Section 3.3 gives an overview of few implementation issues, public-key methods and hierarchical revocation, as well as applications to copy protection and secure multicast. Section 4 provides a traitors-tracing algorithm that works for every revocation algorithm in the Subset-Cover framework and an improvement specifically suited for the Subset-Difference revocation algorithm.

2 The Subset-Cover Revocation Framework

2.1 Preliminaries - Problem Definition

Let $\mathcal{N}$ be the set of all users, $|\mathcal{N}| = N$, and $\mathcal{R} \subset \mathcal{N}$ be a group of $|\mathcal{R}| = r$ users whose decryption privileges should be revoked. The goal of a revocation algorithm is to allow a center to transmit a message M to all users such that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it. The definition of the latter is provided in Sect. 2.3.

A system consists of three parts: (1) An initiation scheme, which is a method for assigning the receivers secret information that will allow them to decrypt. (2) The broadcast algorithm - given a message M and the set $\mathcal{R}$ of users that should be revoked outputs a ciphertext message M' that is broadcast to all receivers. (3) A decryption algorithm - a (non-revoked) user that receives ciphertext M' using its secret information should produce the original message M. Since the receivers are stateless, the output of the decryption should be based on the current message and the secret information only.

2.2 The Framework

We present a framework for algorithms which we call *Subset-Cover*. In this framework an algorithm defines a collection of subsets $S_1, \ldots, S_w$, $S_j \subseteq \mathcal{N}$. Each subset S_j is assigned (perhaps implicitly) a long-lived key L_j; each member u of S_j should be able to deduce L_j from its secret information. Given a revoked set $\mathcal{R}$, the remaining users $\mathcal{N} \setminus \mathcal{R}$ are partitioned into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$ so that $\mathcal{N} \setminus \mathcal{R} = \bigcup_{j=1}^{m} S_{i_j}$ and a session key K is encrypted m times with $L_{i_1}, \ldots, L_{i_m}$.

Specifically, an algorithm in the framework uses two encryption schemes:

- A method $F_K : \{0,1\}^* \mapsto \{0,1\}^*$ to encrypt the message itself. The key K used will be chosen fresh for each message M - a session key - as a random bit string. F_K should be a fast method and should not expand the plaintext. The simplest implementation is to Xor the message M with a stream cipher generated by K.
- A method to deliver the session key to the receivers, for which we will employ an encryption scheme. The keys L here are long-lived. The simplest implementation is to make $E_L : \{0,1\}^\ell \mapsto \{0,1\}^\ell$ a block cipher.

A discussion of the security requirements of these primitives is given in Sect. 2.3. Suggestions for the implementation of F_K and E_L are outlined in Sect. 3.3 and given in [21]. The algorithm consists of three components:

Scheme Initiation: Every receiver u is assigned private information I_u. For all $1 \leq i \leq w$ such that $u \in S_i$, I_u allows u to deduce the key L_i corresponding to the set S_i. Note that the keys L_i can be chosen either (i) uniformly at random and independently from each other (which we call the *information-theoretic* case) or (ii) as a function of other (secret) information (which we call the *computational* case), and thus may not be independent of each other.

The **Broadcast algorithm** at the Center: The center chooses a session encryption key K. Given a set $\mathcal{R}$ of revoked receivers, it finds a partition $S_{i_1}, \ldots, S_{i_m}$ covering all users in $\mathcal{N} \setminus \mathcal{R}$. Let $L_{i_1}, \ldots, L_{i_m}$ be the keys associated with the above subsets. The center encrypts K with keys $L_{i_1}, \ldots, L_{i_m}$ and sends the ciphertext

$$\langle [i_1, i_2, \ldots, i_m, E_{L_{i_1}}(K), E_{L_{i_2}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M) \rangle$$

The portion in square brackets preceding $F_K(M)$ is called the *header* and $F_K(M)$ is called the *body.*
The **Decryption step** at the receiver u, upon receiving a broadcast message $\langle[i_1, i_2, \ldots, i_m, C_1, C_2, \ldots, C_m], M'\rangle$: the receiver finds i_j such that $u \in S_{i_j}$ (in case $u \in \mathcal{R}$ the result is **null**). It then extracts the corresponding key L_{i_j} from I_u, computes $D_{L_{i_j}}(C_j))$ to obtain K and computes $D_K(M')$ to obtain and output M.

A particular implementation of such scheme is specified by (1) the collection of subsets $S_1, \ldots, S_w$ (2) the key assignment to each subset in the collection (3) a method to cover the non-revoked receivers $\mathcal{N} \setminus \mathcal{R}$ by disjoint subsets from this collection, and (4) A method that allows each user u to find its cover S and compute its key L_S from I_u. The algorithm is evaluated based upon three parameters:

1. Message Length - the length of the header that is attached to $F_K(M)$, which is proportional to m, the number of sets in the partition covering $\mathcal{N} \setminus \mathcal{R}$.
2. Storage size at the receiver - how much private information (typically, keys) does a receiver need to store. For instance, I_u could simply consists of all the keys S_i such that $u \in S_i$, or if the key assignment is more sophisticated it should allow the computation of all such keys.
3. Message processing time at receiver. We often distinguish between decryption and other types of operations.

It is important to characterize the dependence of the above three parameters in both N and r. Specifically, we say that a revocation scheme is *flexible with respect to* r if the storage at the receiver is not a function of r. Note that the efficiency of setting up the scheme and computing the partition (given $\mathcal{R}$) is not taken into account in the algorithm's analysis. However, for all schemes presented in this paper the computational requirements of the sender are rather modest: finding the partition takes time linear in $|\mathcal{R}| \log N$ and the encryption is proportional to the number of subsets in the partition. In this framework we demonstrate the substantial gain that can be achieved by using a computational key-assignment scheme as opposed to an information-theoretic one [4].

2.3 Security of the framework

The definition of the Subset-Cover framework allows a rigorous treatment of the security of any algorithm in this family. Unfortunately, due to lack of space, this discussion must be omitted and is included in the full version of the paper [21]. A summary of this analysis follows.

Our contribution is twofold. We first define the notion of revocation-scheme security, namely specify the adversary's power in this scenario and what is considered a successful break. This roughly corresponds to an adversary that may pool the secret information of several users and may have some influence on the

[4] Note that since the assumptions on the security of the encryption primitives are computational, a computational key-assignment method is a natural.

choice of messages encrypted in this scheme (chosen plaintext). Also it may create bogus messages and see how legitimate users (that will not be revoked) react. Finally, to say that the adversary has broken the scheme means that when the users who have provided it their secret information are all revoked (otherwise it is not possible to protect the plaintext) the adversary can still learn something about the encrypted message. Here we define "learn" as distinguishing its encryption from random (this is equivalent to semantic security).

Second, we state the security assumptions on the primitives used in the scheme (these include the encryptions primitives E_L and F_K and the key assignment method in the subset-cover algorithm.) We identify a critical property that is required from the key-assignment method: a subset-cover algorithm satisfies the "key-indistinguishability" property if for every subset S_i its key L_i is *indistinguishable from a random key* given all the information of all users that are *not* in S_i. Note that any scheme in which the keys to all subsets are chosen independently (trivially) satisfies this property. To obtain our security theorem, we require two different sets of properties from E_L and F_K, since F_K uses short lived keys whereas E_L uses long-lived ones. Specifically, E_L is required to be semantically secure against chosen ciphertext attacks in the pre-processing mode, and F_K to be chosen-plaintext, one-message semantically secure (see [21] for details). We then proceed to show that if the subset-cover algorithm satisfies the key-indistinguishability property and if E_L and F_K satisfy their security requirements, then the revocation scheme is secure under the above definition.

Theorem 1. *Let $\mathcal{A}$ be a Subset-Cover revocation algorithm where (i) the key assignment satisfies the key-indistinguishability property (ii) E_L is semantically secure against chosen ciphertext attacks in the pre-processing mode, and (iii) F_K is chosen-plaintext, one-message semantically secure. Then $\mathcal{A}$ satisfies the notion of revocation scheme security defined above.*

3 Two Subset-Cover Revocation Algorithms

We describe two schemes in the Subset-Cover framework with a different performance tradeoff, as summarized in table 1.2[5]. Each is defined over a different collection of subsets. Both schemes are r-flexible, namely they work with any number of revocations. In the first scheme, the key assignment is information-theoretic whereas in the other scheme the key assignment is computational. While the first method is relatively simple, the second method is more involved, and exhibits a *substantial improvement over previous methods.*

In both schemes the subsets and the partitions are obtained by imagining the receivers as the leaves in a rooted full binary tree with N leaves (assume that N is a power of 2). Such a tree contains $2N-1$ nodes (leaves plus internal nodes) and for any $1 \leq i \leq 2N-1$ we assume that v_i is a node in the tree. We denote

[5] Recently a method exhibiting various tradeoffs between the measures (bandwidth, storage and processing time) was proposed [22]. In particular it is possible to reduce the device storage down to $\log^2 n/\log D$ by increasing processing time to $D \log n$.

by $ST(\mathcal{R})$ the (directed) Steiner Tree induced by the set $\mathcal{R}$ of vertices and the root, i.e. the minimal subtree of the full binary tree that connects all the leaves in $\mathcal{R}$ ($ST(\mathcal{R})$ is unique). The systems differ in the collections of subsets they consider.

3.1 The Complete Subtree Method

The collection of subsets $S_1, \ldots, S_w$ in our first scheme corresponds to all complete subtrees in the full binary tree with N leaves. For any node v_i in the full binary tree (either an internal node or a leaf, $2N-1$ altogether) let the subset S_i be the collection of receivers u that correspond to the leaves of the subtree rooted at node v_i. The key assignment method simply assigns an independent and random key L_i to every node v_i in the complete tree. Provide every receiver u with the $\log N + 1$ keys associated with the nodes along the path from the root to leaf u.

For a given set $\mathcal{R}$ of revoked receivers, let $S_{i_1}, \ldots, S_{i_m}$ be all the subtrees of the original tree whose roots are adjacent to nodes of outdegree 1 in $ST(\mathcal{R})$, but they are not in $ST(\mathcal{R})$. It follows immediately that this collection covers all nodes in $\mathcal{N} \setminus \mathcal{R}$ and only them. The cover size is at most $r \log(N/r)$. This is also the average number of subsets in the cover.

At decryption, given a message $\langle [i_1, \ldots, i_m, E_{L_{i_1}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M)] \rangle$ a receiver u needs to find whether any of its ancestors is among $i_1, i_2, \ldots i_m$; note that there can be only one such ancestor, so u may belong to at most one subset. This lookup can be facilitated efficiently by using hash-table lookups with *perfect hash functions*.

The key assignment in this method is information theoretic, that is keys are assigned randomly and independently. Hence the "key-indistinguishability" property of this method follows from the fact that no $u \in \mathcal{R}$ is contained in any of the subsets $i_1, i_2, \ldots i_m$.

Theorem 2. *The Complete Subtree Revocation method requires (i) message length of at most $r \log \frac{N}{r}$ keys (ii) to store $\log N$ keys at a receiver and (iii) $O(\log\log N)$ operations plus a* single *decryption operation to decrypt a message. Moreover, the method is secure in the sense of the definition outlined in 2.3.*

Comparison to the Logical Key Hierarchy (LKH) approach: Readers familiar with the LKH method of [29, 30] may find it instructive to compare it to the Complete Subtree Scheme. The main similarity lies in the key assignment - an independent label is assigned to each node in the binary tree. However, these labels are used quite differently - in the multicast re-keying LKH scheme some of these labels change at every revocation. In the Complete Subtree method labels are *static*; what changes is a single session key.

Consider an extension of the LKH scheme which we call the *clumped re-keying method*: here, r revocations are performed at a time. For a batch of r revocations, no label is changed more than once, i.e. only the "latest" value is transmitted and used. In this variant the number of encryptions is roughly the same as in the Complete Subtree method, but it requires $\log N$ decryptions at the user, (as opposed to a single decryption in our framework). An additional advantage of the Complete Subtree method is the separation of the labels and the session key which has a consequence on the message length; see discussion about Prefix-Truncation in [21].

3.2 The Subset Difference Method

The main disadvantage of the Complete Subtree method is that $\mathcal{N} \setminus \mathcal{R}$ may be partitioned into a number of subsets that is too large. The goal is now to reduce the partition size. We show an improved method that partitions the non-revoked receivers into at most $2r-1$ subsets (or $1.25r$ on average), thus getting rid of a $\log N$ factor and effectively reducing the message length accordingly. In return, the number of keys stored by each receiver increases by a factor of $\frac{1}{2} \cdot \log N$. The key characteristic of the Subset-Difference method, which essentially leads to the reduction in message length, is that in this method any user belongs to *substantially* more subsets than in the first method ($O(N)$ instead of $\log N$). The challenge is then to devise an efficient procedure to succinctly encode this large set of keys at the user, which is achieved by using a computational key assignment.

The subset description As in the previous method, the receivers are viewed as leaves in a complete binary tree. The collection of subsets $S_1, \ldots, S_w$ defined by this algorithm corresponds to subsets of the form "a group of receivers G_1 minus another group G_2", where $G_2 \subset G_1$. The two groups G_1, G_2 correspond to leaves in two full binary subtrees. Therefore a valid subset S is represented by two nodes in the tree (v_i, v_j) such that v_i is an ancestor of v_j. We denote such subset as $S_{i,j}$. A leaf u is in $S_{i,j}$ iff it is in the subtree rooted at v_i but *not* in the subtree rooted at v_j, or in other words $u \in S_{i,j}$ iff v_i is an ancestor of u but v_j is not. Figure 2 depicts $S_{i,j}$. Note that all subsets from the Complete Subtree Method are also subsets of the Subset Difference Method; specifically, a subtree appears here as the difference between its parent and its sibling. The only exception is the full tree itself, and we will add a special subset for that. We postpone the description of the key assignment till later; for the time being assume that each subset $S_{i,j}$ has an associated key $L_{i,j}$.

The Cover For a set $\mathcal{R}$ of revoked receivers, the following Cover algorithm finds a collection of disjoint subsets $S_{i_1,j_1}, S_{i_2,j_2} \ldots, S_{i_m,j_m}$ which partitions $\mathcal{N} \setminus \mathcal{R}$. The method builds the subsets collection iteratively, maintaining a tree T which is a subtree of $ST(\mathcal{R})$ with the property that any $u \in \mathcal{N} \setminus \mathcal{R}$ that is below a leaf of T has been covered. We start by making T be equal to $ST(\mathcal{R})$ and then iteratively remove nodes from T (while adding subsets to the collection) until T consists of just a single node:

1. Find two leaves v_i and v_j in T such that the least-common-ancestor v of v_i and v_j does not contain any other leaf of T in its subtree. Let v_l and v_k be

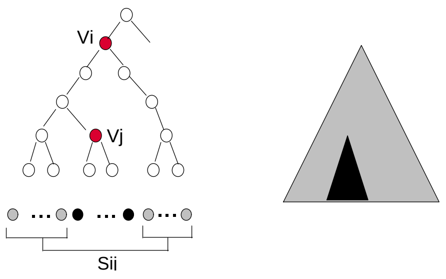

Fig. 2. The Subset Difference method: subset $S_{i,j}$ contains all marked leaves (non-black).

the two children of v such that v_i a descendant of v_l and v_j a descendant of v_k. (If there is only one leaf left, make $v_i = v_j$ to the leaf, v to be the root of T and $v_l = v_k = v$.)
2. If $v_l \not\equiv v_i$ then add the subset $S_{l,i}$ to the collection; likewise, if $v_k \not\equiv v_j$ add the subset $S_{k,j}$ to the collection.
3. Remove from T all the descendants of v and make it a leaf.

An alternative description of the cover algorithm is as follows: consider maximal chains of nodes with outdegree 1 in $ST(\mathcal{R})$. More precisely, each such chain is of the form $[v_{i_1}, v_{i_2}, \ldots v_{i_\ell}]$ where (i) all of $v_{i_1}, v_{i_2}, \ldots v_{i_{\ell-1}}$ have outdegree 1 in $ST(\mathcal{R})$ (ii) v_{i_ℓ} is either a leaf or a node with outdegree 2 and (iii) the parent of v_{i_1} is either a node of outdegree 2 or the root. For each such chain where $\ell \geq 2$ add a subsets S_{i_1,i_ℓ} to the cover. Note that all nodes of outdegree 1 in $ST(\mathcal{R})$ are members of precisely one such chain.

We state, without a proof, that a cover can contain at most $2r - 1$ subsets for any set of r revocations. Moreover, if the set of revoked leaves is *random*, then average-case analysis bounds the cover size by $1.38r$, whereas simulation experiments tighten the bound to $1.25r$.

The next lemma is concerned with covering more general sets than those obtained by removing users. Rather it assumes that we are removing a collection of subsets from the Subset Difference collection. It is applied later in Section 4.2.

Lemma 1. *Let $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$ be a collection of m* **disjoint** *subsets from the underlying collection defined by the Subset Difference method, and $\mathcal{U} = \cup_{j=1}^{m} S_{i_j}$. Then the leaves in $\mathcal{N} \setminus \mathcal{U}$ can be covered by at most $3m - 1$ subsets from the underlying Subset Difference collection.*

Key assignment to the subsets We now define what information each receiver must store. If we try and repeat the information-theoretic approach of the previous scheme where each receiver needs to store *explicitly* the keys of all the subsets it belongs to, the storage requirements would expand tremendously: consider a receiver u; for each complete subtree T_k it belongs to, u must store a number of keys proportional to the number of nodes in the subtree T_k that are *not* on the path from the root of T_k to u. There are $\log N$ such trees, one for each height $1 \leq k \leq \log N$, yielding a total of $\sum_{k=1}^{\log N}(2^k - k)$ which is $O(N)$ keys. We therefore devise a key assignment method that requires a receiver to store only $O(\log N)$ keys per subtree, for the total of $O(\log^2 N)$ keys.

While the total number of subsets to which a user u belongs is $O(N)$, these can be grouped into $\log N$ clusters defined by the first subset i (from which another subsets is subtracted). The way we proceed with the keys assignment is to choose for each $1 \leq i \leq N - 1$ corresponding to an internal node in the full binary tree a random and independent value LABEL_i. This value should *induce* the keys for all legitimate subsets of the form $S_{i,j}$. The idea is to employ the method used by Goldreich, Goldwasser and Micali [17] to construct pseudo-random functions, which was also used by Fiat and Naor [12] for purposes similar to ours.

Let G be a (cryptographic) pseudo-random sequence generator (see definition below) that *triples* the input, i.e. whose output length is *three times* the length of the input; let $G_L(S)$ denote the left third of the output of G on seed S, $G_R(S)$ the right third and $G_M(S)$ the middle third. We say that $G : \{0,1\}^n \mapsto \{0,1\}^{3n}$ is a pseudo-random sequence generator if no polynomial-time adversary can distinguish the output of G on a randomly chosen seed from a truly random string of similar length. Let ε_4 denote the bound on the distinguishing probability.

Consider now the subtree T_i (rooted at v_i). We will use the following top-down labeling process: the root is assigned a label LABEL_i. Given that a parent was labeled S, its two children are labeled $G_L(S)$ and $G_R(S)$ respectively. Let $\mathrm{LABEL}_{i,j}$ be the label of node v_j derived in the subtree T_i from LABEL_i. Following such a labeling, the key $L_{i,j}$ assigned to set $S_{i,j}$ is G_M of $\mathrm{LABEL}_{i,j}$. Note that each label induces three parts: G_L - the label for the left child, G_R - the label for the right child, and G_M the key at the node. The process of generating labels and keys for a particular subtree is depicted in Fig. 3. For such a labeling process, given the label of a node it is possible to compute the labels (and keys) of all its descendants. On the other hand, without receiving the label of an ancestor of a node, its label is pseudo-random and for a node j, given the labels of all its descendants (but not including itself) the key $L_{i,j}$ is pseudo-random ($\mathrm{LABEL}_{i,j}$, the label of v_j, is not pseudo-random given this information simply because one can check for consistency of the labels). It is important to note that given LABEL_i, computing $L_{i,j}$ requires at most $\log N$ invocations of G.

We now describe the information I_u that each receiver u gets in order to derive the key assignment described above. For each subtree T_i such that u is a leaf of T_i the receiver u should be able to compute $L_{i,j}$ iff j is *not* an ancestor of u. Consider the path from v_i to u and let $v_{i_1}, v_{i_2}, \ldots v_{i_k}$ be the nodes just "hanging off" the path, i.e. they are adjacent to the path but not ancestors of u (see Fig. 3). Each j in T_i that is not an ancestor of u is a descendant of one of these nodes. Therefore if u receives the labels of $v_{i_1}, v_{i_2}, \ldots v_{i_k}$ as part of I_u, then invoking G at most $\log N$ times suffices to compute $L_{i,j}$ for any j that is not an ancestor of u.

Fig. 3. Key assignment in the Subset Difference method. *Left:* generation of $\mathrm{LABEL}_{i,j}$ and the key $L_{i,j}$. *Right:* leaf u receives the labels of $v_{i_1}, \ldots v_{i_k}$ that are induced by the label LABEL_i of v_i.

As for the total number of keys (in fact, labels) stored by receiver u, each tree T_i of depth k that contains u contributes $k-1$ keys (plus one key for the case where there are no revocations), so the total is $1 + \sum_{k=1}^{\log N+1} k - 1 = 1 + \frac{(\log N+1)\log N}{2} = \frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$.

At decryption time, a receiver u first finds the subset $S_{i,j}$ such that $u \in S_{i,j}$, and computes the key corresponding to $L_{i,j}$. Using the techniques described in the complete subtree method for table lookup structure, this subset can be found in $O(\log\log N)$. The evaluation of the subset key takes now at most $\log N$ applications of a pseudo-random generator. After that, a single decryption is needed.

Security In order to prove security we have to show that the key indistinguishability condition (outlined in Sect. 2.3) holds for this method, namely that each key is indistinguishable from a random key for all users not in the corresponding subset.

Observe first that for any $u \in \mathcal{N}$, u never receives keys that correspond to subtrees to which it does not belong. Let S_i denote the set of leaves in the subtree T_i rooted at v_i. For any set $S_{i,j}$ the key $L_{i,j}$ is (information theoretically) independent of all I_u for $u \notin S_i$. Therefore we have to consider only the combined secret information of all $u \in S_j$. This is specified by at most $\log N$ labels - those hanging on the path from v_i to v_j plus the two children of v_j - which are sufficient to derive all other labels in the combined secret information. Note that these labels are $\log N$ strings that were generated independently by G, namely it is never the case that one string is derived from another. Hence, a hybrid argument implies that the probability of distinguishing $L_{i,j}$ from random can be at most $\varepsilon_4/\log N$, where ε_4 is the bound on distinguishing outputs of G from random strings.

Theorem 3. *The Subset Difference method requires (i) message length of at most* $2r - 1$ *keys (ii) to store* $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$ *keys at a receiver and (iii)* $O(\log N)$ *operations plus a* single *decryption operation to decrypt a message. Moreover, the method is secure in the sense of definition outlined in 2.3.*

3.3 Further Discussions (Summary)

In [21] we discuss a number of important issues related to the above schemes, their implementation and applications. Below is a short summary of the topics.

Implementation Issues: A key characteristic of the Subset-Cover framework is that it clearly separates the long-term keys from the short, one time, key. This allows, if so desired, to chose an encryption F that might be weaker (uses shorter keys) than the encryption chosen for E and to reduce the message length appropriately. We provide a "Prefix-Truncation" specification for E to implement such a reduction without sacrificing the security of the long-lived keys. Let $\text{Prefix}_i S$ denote the first i bits of a string S. choose $\mathcal{U}$ to be a random string whose length is the length of the block of E_L and let K be a relatively short key for the cipher F_K (whose length is, say, 56 bits). Then, $[\text{Prefix}_{|K|} E_L(\mathcal{U})] \oplus K$ provides an encryption that satisfies the requirements of E, as described in Sect. 2.3. The Prefix-Truncated header is therefore:

$$\langle [\; i_1, \ldots, i_m, \mathcal{U}, [\text{Prefix}_{|K|} E_{L_{i_1}}(\mathcal{U})] \oplus K, \ldots, [\text{Prefix}_{|K|} E_{L_{i_m}}(\mathcal{U})] \oplus K \;], F_K(M) \rangle$$

Note that the length of the header is reduced to about $m \times |K|$ bits long (say $56m$) instead of $m \times |L|$.

Hierarchical Revocation: We point out that the schemes are well suited to efficiently support hierarchical revocations of large groups of clustered-users; this is useful, for instance, to revoke all devices of a certain manufacturer.

Public Key methods: A revocation scheme that is used in a public key mode is appropriate in scenarios where the the party that generated the ciphertext is not necessarily trustworthy. This calls for implementing E with a public-key cryptosystem; however, a number of difficulties arise such as the public-key generation process, the size of the public key file and the header reduction. As we show, using a Diffie-Hellman like scheme solves most of these problems (except the public key file size).

An interesting point is that prefix truncation is still applicable and we get that the length of public-key encryption is hardly longer than the private-key case. This can be done as follows: Let G be a group with a generator g, $g^{y_{i_j}}$ be the public key of subset S_{i_j} and y_{i_j} the secret key. Choose h as a pairwise-independent function $h : G \mapsto \{0,1\}^{|K|}$, thus elements which are uniformly distributed over G are mapped to uniformly distributed strings of the desired length. The encryption E is done by picking a new element x from G, publicizing g^x, and encrypting K as $E_{L_{i_j}}(K) = h(g^{xy_{i_j}}) \oplus K$. That is, the header now becomes

$$\langle [\; i_1, i_2, \ldots, i_m, g^x, h, \; h(g^{xy_{i_1}}) \oplus K, \ldots, h(g^{xy_{i_m}}) \oplus K \;], F_K(M) \rangle$$

In terms of the broadcast length such system hardly increases the number of bits in the header as compared with a shared-key system - the only difference is g^x and the description of h. Therefore this difference is fixed and does not grow with the number of revocations. Note however that the scheme as defined above is not immune to chosen-ciphertext attacks, but only to chosen plaintext ones. Coming up with public-key schemes where prefix-truncation is possible that are immune to chosen ciphertext attacks of either kind is an interesting challenge[6].

Copy Protection and CPRM Copy protection is a natural application for trace-and-revoke schemes, and the stateless scenario is especially appropriate when content is distributed on pre-recorded media. CPRM/CPPM (Content Protection for Recordable Media and Pre-Recorded Media) is a technology developed and licensed by the "4C" group - IBM, Intel, MEI (Panasonic) and Toshiba [10]. It defines a method for protecting content on physical media such as recordable DVD, DVD Audio, Secure Digital Memory Card and Secure CompactFlash. A licensing Entity (the Center) provides a unique set of secret device keys to be included in each device at manufacturing time. The licensing Entity also provides a Media Key Block (MKB) to be placed on each compliant media (for example, on the DVD). The MKB is essentially the header of the ciphertext which encrypts the session key. It is assumed that this header resides on a write-once area on the media, e.g. a Pre-embossed lead-in area on the recordable DVD. When the compliant media is placed in a player/recorder device, it computes the session key from the header (MKB) using its secret keys; the content is then encrypted/decrypted using this session key.

The algorithm employed by CPRM is essentially a Subset-Cover scheme. Consider a table with A rows and C columns. Every device (receiver) is viewed as a collection of C entries from the table, exactly one from each column, that is $u = [u_1, \ldots, u_C]$ where $u_i \in \{0, 1, \ldots, A-1\}$. The collection of subsets $S_1, \ldots, S_w$ defined by this algorithm correspond to subsets of receivers that share the same entry at a given column, namely $S_{r,i}$ contains all receivers $u = [u_1, \ldots, u_C]$ such that $u_i = r$. For every $0 \leq i \leq A-1$ and $1 \leq j \leq C$ the scheme associates a key denoted by $L_{i,j}$. The private information I_u that is provided to a device $u = [u_1, \ldots, u_C]$ consists of C keys $L_{u_1,1}, L_{u_2,2}, \ldots, L_{u_C,C}$.

For a given set $\mathcal{R}$ of revoked devices, the method partitions $\mathcal{N} \setminus \mathcal{R}$ as follows: $S_{i,j}$ is in the cover iff $S_{i,j} \bigcap \mathcal{R} = \emptyset$. While this partition guarantees that a revoked device is never covered, there is a low probability that a non-revoked device $u \notin \mathcal{R}$ will not be covered as well and therefore become non-functional[7].

The CPRM method is a Subset-Cover method with two exceptions: (1) the subsets in a cover are not necessarily disjoint and (2) the cover is not always perfect as a non-revoked device may be uncovered. Note that the CPRM method is not *r-flexible*: the probability that a non-revoked device is uncovered grows

[6] Both the scheme of Cramer and Shoup [7] and the random oracle based scheme [14] require some specific information for each recipient; a possible approach with random oracles is to follow the lines of [27].

[7] This is similar to the scenario considered in [16]

with r, hence in order to keep it small enough the number of revocations must be bounded by A.

For the sake of comparing the performance of CPRM with the two methods suggested in this paper, assume that $C = \log N$ and $A = r$. Then, the message is composed of $r \log N$ encryptions, the storage at the receiver consists of $\log N$ keys and the computation at the receiver requires a single decryption. These bounds are similar to the Complete Subtree method; however, unlike CPRM, the Complete Subtree method is r-flexible and achieves perfect coverage. The advantage of the Subset Difference Method is much more substantial: in addition to the above, the message consists of $1.25r$ encryptions on average, or of at most $2r - 1$ encryptions, rather than $r \log N$.

For example, in DVD Audio, the amount of storage that is dedicated for its MKB (the header) is 3 MB. This constrains the maximum allowed message length. Under a certain choice of parameters, such as the total number of manufactured devices and the number of distinct manufacturers, with the current CPRM algorithm the system can revoke up to about 10,000 devices. In contrast, for the same set of parameters and the same 3MB constraint, a Subset-Difference algorithm achieves up to 250,000 (!) revocations, a factor of 25 improvement over the currently used method. This major improvement is partly due to fact that hierarchical revocation can be done very effectively, a property that the current CPRM algorithm does not have.

Applications to Multicast The difference between key management for the scenario considered in this paper and for the Logical Key Hierarchy for multicast is that in the latter the users (i.e. receivers) may update their keys [30, 29]. This update is referred to as a re-keying event and it requires all users to be connected during this event and change their internal state (keys) accordingly. However, even in the multicast scenario it is not reasonable to assume that all the users receive all the messages and perform the required update. Therefore some mechanism that allows individual update must be in place. Taking the stateless approach gets rid of the need for such a mechanism: simply add a header to each message denoting who are the legitimate recipients by revoking those who should not receive it. In case the number of revocations is not too large this may yield a more manageable solution. This is especially relevant when there is a single source for the sending messages or when public-keys are used.

Backward secrecy: Note that revocation in itself lacks backward secrecy in the following sense: a constantly listening user that has been revoked from the system records all future transmission (which it can't decrypt anymore) and keeps all ciphertexts. At a later point it gains a valid *new* key (by re-registering) which allows decryption of all past communication. Hence, a newly acquired user-key can be used to decrypt all past session keys and ciphertexts. The way that [30, 29] propose to achieve backward secrecy is to perform re-keying when new users are added to the group (such a re-keying may be reduced to only one way chaining, known as LKH+), thus making such operations non-trivial. We point out that in the subset-cover framework and especially in the two methods we proposed it may be easier: At any given point of the system include in the set of revoked receivers all identities that have not been assigned yet. As a result, a newly assigned user-key cannot help in decrypting an earlier ciphertext. Note that this is feasible since we assume that new users are assigned keys in a consecutive order of the leaves in the tree, so unassigned keys are consecutive leaves in the complete tree and can be covered by at most $\log N$ sets (of either type, the Complete-Subtree method or the Subtree-Difference method). Hence, the unassigned leaves can be treated with the hierarchical revocation technique, resulting in *adding* at most $\log N$ revocations to the message.

4 Tracing Traitors

It is highly desirable that a revocation mechanism could work in tandem with a tracing mechanism to yield a *trace and revoke* scheme. We show a tracing method that works for many schemes in the subset-cover framework. The method is quite efficient. The goal of a tracing algorithm is to find the identities of those that contributed their keys to an illicit decryption box[8] and revoke them; short of identifying them we should render the box useless by finding a "pattern" that does not allow decryption using the box, but still allows broadcasting to the legitimate users. Note that this is a slight relaxation of the requirement of a tracing mechanism, say in [23] (which requires an identification of the traitor's identity) and in particular it lacks *self enforcement* [11]. However as a mechanism that works in conjunction with the revocation scheme it is a powerful tool to combat piracy.

The model Suppose that we have found an illegal decryption-box (decoder, or clone) which contains the keys associated with at most t receivers $u_1, \ldots, u_t$ known as the "traitors".

We are interested in "black-box" tracing, i.e. one that does not take the decoder apart but by providing it with an encrypted message and observing its output (the decrypted message) tries to figure out who leaked the keys. A pirate decoder is of interest if it correctly decodes with probability p which is at least some threshold q, say $q > 0.5$. We assume that the box has a "reset button", i.e. that its internal state may be retrieved to some initial configuration. In particular this excludes a "locking" strategy on the part of the decoder which says that in case it detects that it is under test, it should refuse to decode further. Clearly software-based systems can be simulated and therefore have the reset property.

The result of a tracing algorithm is either a subset consisting of traitors or a partition into subsets that renders the box useless i.e. given an encryption with the given partition it decrypts with probability smaller than the threshold q while all good users can still decrypt.

In particular, a "subsets based" tracing algorithm devises a sequence of queries which, given a black-box that decodes with probability above the threshold q, produces the results mentioned above. It is based on constructing useful

[8] Our algorithm also works for more than one box.

sets of revoked devices $\mathcal{R}$ which will ultimately allow the detection of the receiver's identity or the configuration that makes the decoder useless. A tracing algorithm is evaluated based on (i) the level of performance downgrade it imposes on the revocation scheme (ii) number of queries needed.

4.1 The Tracing Algorithm

Subset tracing: An important procedure in our tracing mechanism is one that given a partition $\mathcal{S} = S_{i_1}, S_{i_2}, \dots S_{i_m}$ and an illegal box outputs one of two possible outputs: either (1) that the box cannot decrypt with probability greater than the threshold when the encryption is done with partition $\mathcal{S}$ or (ii) Finds a subset S_{i_j} such that S_{i_j} contains a traitor. Such a procedure is called subset tracing and is described below.

Bifurcation property: Given a subset-tracing procedure, we describe a tracing strategy that works for many Subset-Cover revocation schemes. The property that the revocation algorithm should satisfy is that for any subset $S_i, 1 \leq i \leq w$, it is possible to partition S_i into two (or constant) roughly equal sets, i.e. that there exists $1 \leq i_1, i_2 \leq w$ such that $S_i = S_{i_1} \cup S_{i_2}$ and $|S_{i_1}|$ is roughly the same as $|S_{i_2}|$. For a Subset Cover scheme, let the *bifurcation value* be the relative size of the largest subset in such a split.

Both the Complete Subtree and the Subtree Difference methods satisfy this requirement: in the case of the Complete Subtree Method each subset, which is a complete subtree, can be split into exactly two equal parts, corresponding to the left and right subtrees. Therefore the bifurcation value is 1/2. As for the Subtree Difference Method, Each subset $S_{i,j}$ can be split into two subsets each containing between one third and two thirds of the elements. Here, again, this is done using the left and right subtrees of node i. See Fig. 4. The only exception is when i is a parent of j, in which case the subset is the complete subtree rooted at the other child; such subsets can be perfectly split. The worst case of $(1/3, 2/3)$ occurs when i is the grandparent of j. Therefore the bifurcation value is 2/3.

The Tracing Algorithm: We now describe the general tracing algorithm, assuming that we have a good subset tracing procedure. The algorithm maintains a partition $S_{i_1}, S_{i_2}, \dots S_{i_m}$. At each phase one of the subsets is partitioned, and the goal is to partition a subset only if it contains a traitor.

Each phase initially applies the subset-tracing procedure with the current partition $\mathcal{S} = S_{i_1}, S_{i_2}, \dots S_{i_m}$. If the procedure outputs that the box cannot decrypt with $\mathcal{S}$ then we are done, in the sense that we have found a way to disable the box without hurting any legitimate user. Otherwise, let S_{i_j} be the set output by the procedure, namely S_{i_j} contains the a traitor.

If S_{i_j} contains only one possible candidate - it must be a traitor and we permanently revoke this user; this doesn't hurt a legitimate user. Otherwise we split S_{i_j} into two roughly equal subsets and continue with the new partitioning. The existence of such a split is assured by the bifurcation property.

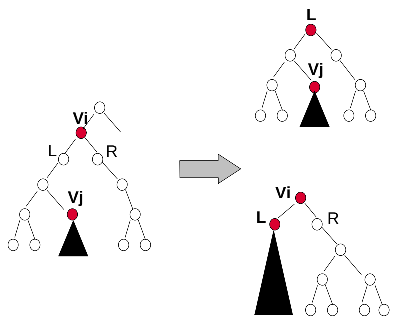

Fig. 4. Bifurcating a Subset Difference set $S_{i,j}$, depicted in the left. The black triangle indicates the excluded subtree. L and R are the left and the right children of v_i. The resulting sets $S_{L,j}$ and $S_{i,L}$ are depicted to the right.

Analysis: Since a partition can occur only in a subset that has a traitor and contains more than one element, it follows that the number of iterations can be at most $t \log_a N$, where a is the inverse of the bifurcation value (a more refined expression is $t(\log_a N - log_2 t)$, the number of edges in a binary tree with t leaves and depth $\log_a N$.)

The Subset Tracing Procedure The Subset Tracing procedure first tests whether the box decodes a message with the partition $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$ with sufficient probability greater than the threshold, say > 0.5. If not, then it concludes (and outputs) that the box cannot decrypt with $\mathcal{S}$. Otherwise, it needs to find a subset S_{i_j} that contains a traitor.

Let p_j be the probability that the box decodes the ciphertext

$$\langle [i_1, i_2, \ldots, i_m, E_{L_{i_1}}(R_K), \ldots, E_{L_{i_j}}(R_K), E_{L_{i_{j+1}}}(K), \ldots, E_{L_{i_m}}(K)], F_K(M) \rangle$$

where R_K is a random string of the same length as the key K. That is, p_j is the probability of decoding when the first j subsets are noisy and the remaining subsets encrypt the correct key. Note that $p_0 = p$ and $p_m = 0$, hence there must be some $0 < j \leq m$ for which $|p_{j-1} - p_j| \geq \frac{p}{m}$. It can be shown that if p_{j-1} is different from p_j by more than ε, where ε is an upper bound on the sum of the probabilities of breaking the encryption scheme E and key assignment method, then the set S_{i_j} must contain a traitor. It also provides a binary-search-like method that efficiently finds a pair of values p_j, p_{j-1} among $p_0, \ldots, p_m$ satisfying $|p_{j-1} - p_j| \geq \frac{p}{m}$.

4.2 Improving the Tracing Algorithm

The basic traitors tracing algorithm described above requires $t \log(N/t)$ iterations. Furthermore, since at each iteration the number of subsets in the partition increases by one, tracing t traitors may result with up to $t \log(N/t)$ subsets and hence in messages of length $t \log(N/t)$. This bound holds for any Subset-Cover method satisfying the *Bifurcation property*, and both the Complete Subtree and the Subset Difference methods satisfy this property. What is the bound on the number of traitors that the algorithm can trace?

Recall that the message length required by the Complete Subtree method is $r \log(N/r)$ for r revocations, hence the tracing algorithm can trace up to r traitors if it uses the Complete Subtree method. However, since the message length of the Subset Difference method is at most $2r-1$, only $\frac{2r-1}{\log N/r}$ traitors can be traced if Subset Difference is used. We now describe an improvement on the basic tracing algorithm that reduces the number of subsets in the partition to $5t-1$ for the Subset Difference method (although the number of iterations remains $t \log(N/t)$). With this improvement the algorithm can trace up to $r/5$ traitors.

Note that among the $t \log N/t$ subsets generated by the basic tracing algorithm, only t actually contain a traitor. The idea is to repeatedly merge those subsets which are not known to contain a traitor.[9] Specifically, we maintain at each iteration a *frontier* of at most $2t$ subsets plus $3t-1$ additional subsets. In the following iteration a subset that contains a traitor is further partitioned; as a result, a new *frontier* is defined and the remaining subsets are re-grouped.

Frontier subsets: Let $S_{i_1}, S_{i_2}, \ldots S_{i_m}$ be the partition at the current iteration. A pair of subsets $(S_{i_{j1}}, S_{i_{j2}})$ is said to be in the frontier if $S_{i_{j1}}$ and $S_{i_{j2}}$ resulted from a split-up of a single subset at an earlier iteration. Also neither ($S_{i_{j1}}$ nor $S_{i_{j2}}$) was singled out by the subset tracing procedure so far. This definition implies that the frontier is composed of k disjoint pairs of *buddy subsets*. Since buddy-subsets are disjoint and since each pair originated from a single subset that contained a traitor (and therefore has been split) $k \leq t$.

We can now describe the improved tracing algorithm which proceeds in iterations. Every iteration starts with a partition $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$. Denote by $F \subset S$ the frontier of S. An iteration consists of the following steps, by the end of which a new partition $\mathcal{S}'$ and a new frontier F' is defined.

- As before, use the Subset Tracing procedure to find a subset S_{i_j} that contains a traitor. If the tracing procedure outputs that the box can not decrypt with $\mathcal{S}$ then we are done. Otherwise, split S_{i_j} into $S_{i_{j1}}$ and $S_{i_{j2}}$.
- $F' = F \cup S_{i_{j1}} \cup S_{i_{j2}}$ ($S_{i_{j1}}$ and $S_{i_{j2}}$ are now in the frontier). Furthermore, if S_{i_j} was in the frontier F and S_{i_k} was its buddy-subset in F then $F' = F' \setminus S_{i_k}$ (remove S_{i_k} from the frontier).
- Compute a cover $\mathcal{C}$ for all receivers that are not covered by F'. Define the new partition $\mathcal{S}'$ as the union of $\mathcal{C}$ and F'.

To see that the process described above converges, observe that at each iteration the number of new *small* frontier sets always increases by at least one. More

[9] This idea is similar to the second scheme of [13], Sect. 3.3. However, in [13] the merge is straightforward as their model allows any subset. In our model only members from the Subset Difference are allowed, hence a merge which produces subsets of this particular type is non-trivial.

precisely, at the end of each iteration construct a vector of length N describing how many sets of size i, $1 \leq i \leq N$, constitute the frontier. It is easy to see that these vectors are lexicographically increasing. The process must stop when or before all sets in the frontier are singletons.

By definition, the number of subsets in a frontier can be at most $2t$. Furthermore, they are paired into at most t disjoint buddy subsets. As for non-frontier subsets ($\mathcal{C}$), Lemma 1 shows that covering the remaining elements can be done by at most $|F| \leq 3t - 1$ subsets (note that we apply the lemma so as to cover all elements that are not covered by the buddy subsets, and there are at most t of them). Hence the partition at each iteration is composed of at most $5t - 1$ subsets.

Acknowledgements

We thank Omer Horvitz for many comments regarding the paper and the implementation of the system. We thank Ravi Kumar, Nelly Fazio, Florian Pestoni and Victor Shoup for useful comments.

References

1. J. Anzai, N. Matsuzaki and T. Matsumoto, A Quick Group Key Distribution Sceheme with "Entity Revocation", Advances in Cryptology - Asiacrypt '99, LNCS 1716, Springer, 1999, pp. 333–347.
2. O. Berkman, M. Parnas and J. Sgall, Efficient Dynamic Traitor Tracing, Proc. of the 11th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 586–595, 2000.
3. D. Boneh and M. Franklin, An efficient public key traitor tracing scheme, Advances in Cryptology - Crypto '99, LNCS 1666, Springer, 1999, pp. 338–353.
4. D. Boneh, and J. Shaw, *Collusion Secure Fingerprinting for Digital Data*, IEEE Transactions on Information Theory, Vol 44, No. 5, pp. 1897–1905, 1998.
5. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, Multicast Security: A Taxonomy and Some Efficient Constructions, Proc. of INFOCOM '99, Vol. 2, pp. 708–716, New York, NY, March 1999.
6. R. Canetti, T. Malkin, K. Nissim, Efficient Communication-Storage Tradeoffs for Multicast Encryption, Advances in Cryptology - EUROCRYPT '99, LNCS 1592, Springer, 1999, pp. 459–474.
7. R. Cramer and V. Shoup, A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. Advances in Cryptology - CRYPTO 1999, Lecture Notes in Computer Science 1462, Springer, pp. 13–25.
8. B. Chor, A. Fiat and M. Naor, Tracing traitors, Advances in Cryptology - CRYPTO '94, LNCS 839, Springer, pp. 257–270, 1994.
9. B. Chor, A. Fiat, M. Naor and B. Pinkas, Tracing traitors, IEEE Transactions on Information Theory, Vol. 46, No. 3, May 2000.
10. Content Protection for Recordable Media. Available: http://www.4centity.com/4centity/tech/cprm
11. C. Dwork, J. Lotspiech and M. Naor, Digital Signets: Self-Enforcing Protection of Digital Information, 28th Symp. on the Theory of Computing, 1996, pp. 489– 498.
12. A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - CRYPTO '93, LNCS 773, Springer, 1994, pp. 480—491.

13. A. Fiat and T. Tassa, *Dynamic Traitor Tracing* Advances in Cryptology - CRYPTO '99, LNCS 1666, 1999, pp. 354–371.
14. E. Fujisaki and T. Okamoto, *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, Advances in Cryptology - CRYPTO 1999, LNCS 1666, 1999, pp. 537–554.
15. E. Gafni, J. Staddon and Y. L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*, Advances in Cryptology - CRYPTO'99, LNCS 1666, Springer, 1999, pp. 372–387.
16. J.A. Garay, J. Staddon and A. Wool, Long-Lived Broadcast Encryption. Advances in Cryptology - CRYPTO'2000, LNCS 1880, pp. 333–352, 2000.
17. O. Goldreich, S. Goldwasser and S. Micali, How to Construct Random Functions. JACM 33(4): 792–807 (1986)
18. R. Kumar, R. Rajagopalan and A. Sahai, Coding Constructions for blacklisting problems without Copmutational Assumptions. Advances in Cryptology - CRYPTO '99, LNCS 1666, 1999, pp. 609–623.
19. M. Luby and J. Staddon, Combinatorial Bounds for Broadcast Encryption. Advances in Cryptology - EUROCRYPT '98, LNCS vol 1403, 1998, pp. 512–526.
20. D. McGrew, A. T. Sherman, *Key Establishment in Large Dynamic Groups Using One-Way Function Trees*, submitted to IEEE Transactions on Software Engineering (May 20, 1998).
21. D. Naor, M. Naor, J. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers*, full version available at the IACR Crypto Archive http://eprint.iacr.org/.
22. M. Naor, *Tradeoffs in Subset-Cover Revocation Schemes*, manuscript, 2001.
23. M. Naor and B. Pinkas, Threshold traitor tracing, Advances in Cryptology - Crypto '98, LNCS 1462, pp. 502–517.
24. M. Naor and B. Pinkas, Efficient Trace and Revoke Schemes Financial Cryptography '2000, LNCS , Springer.
25. B. Pfitzmann, Trials of Traced Traitors, Information Hiding Workshop, First International Workshop, Cambridge, UK, LNCS 1174, Springer, 1996, pp. 49–64.
26. R. Safavi-Naini and Y. Wang, Sequential Traitor Tracing, Advances in Cryptology - CRYPTO 2000, LNCS 1880, pp. 316–332, 2000.
27. V. Shoup and R. Gennaro, Securing threshold cryptosystems against chosen ciphertext attack, Advances in Cryptology - EUROCRYPT '98, LNCS 1403, 1998, pp. 1–16.
28. D.R. Stinson and R. Wei, Key Preassigned Traceability Schemes for Broadcast Encryption, Proc. Fifth Annual Workshop on Selected Areas in Cryptography, LNCS 1556 (1999), pp. 144–156.
29. D.M. Wallner, E.J. Harder and R.C. Agee, *Key Management for Multicast: Issues and Architectures*, Internet Request for Comments 2627, June, 1999. Available: `ftp.ietf.org/rfc/rfc2627.txt`
30. C. K. Wong, M. Gouda and S. Lam, Secure Group Communications Using Key Graphs, Proc. ACM SIGCOMM'98, pp. 68–79.