Revocation and Tracing Schemes for Stateless Receivers*

Dalit Naor[†] Moni Naor[‡] Jeff Lotspiech[§]

June 2002

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 guarantee 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 adapted 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 and (2) providing a seamless integration between the revocation and tracing so that the tracing mechanism does not require any change to the revocation algorithm.

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.

---

*An extended abstract appeared in Crypto 2001.

[†]IBM Haifa Research Lab, Haifa University Campus, Mt. Carmel, Haifa, 31905, Israel. Email: dalit@il.ibm.com. Most of this work was done while at the IBM Almaden Research Center.

[‡]Department of Computer Science and Applied Math, Weizmann Institute, Rehovot Israel. Most of this work was done while visiting IBM Almaden Research Center and Stanford University. Partially supported by DARPA contract F30602-99-1-0530. Email: naor@wisdom.weizmann.ac.il

[§]IBM Almaden Research Center, 650 Harry Road, San-Jose, CA. 95120, USA. Email: lots@almaden.ibm.com

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.

This paper introduces very efficient revocation schemes which are specially 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 $\mathcal{R}$.

Copyright Protection

An important application that motivates the study of revocation and tracing mechanisms is Copyright Protection. The distribution of copyright protected content for (possibly) disconnected operations involves encryption of the content on media. The medium (such as CD, DVD or a flash memory card) typically contains in its header the encryption of the key K which encrypts the content following the header. Compliant devices, or receivers, store appropriate decryption keys that can be used to decrypt the header and in turn decrypt the content. A *copyright protection mechanism* defines the algorithm which assigns keys to devices and encrypts the content.

An essential requirement from a copyright protection mechanism is the ability to revoke, during the lifetime of the system, devices that are being used illegally. It is expected that some devices will be compromised, either via reverse engineering or due to sloppy manufacturing of the devices. As a result keys of a number of compromised devices can then be cloned to form a decrypting unit. This copyright protection violation can be combated by revoking the keys of the compromised devices. Note that devices are stateless as they are assumed to have no capability of dynamically storing any information (other than the original information that is stored at the time of manufacturing) and also since they are typically not connected to the world (except via the media). Hence, it is the responsibility of the media to carry the current state of the system at the time of recording in terms of revoked devices.

It is also highly desirable that the revocation algorithm be coupled with a traitor-tracing mechanism. Specifically, a well-designed copyright protection mechanism should be able to combat piracy in the form of illegal boxes or clone decoding programs. Such decoders typically contain the identities of a *number* of devices that are cooperating; furthermore, they are hard to disassemble[1]. The tracing mechanism should therefore treat the illicit decoder as a black box and simply examine its input/output relationship. A combination of a revocation and a tracing mechanism provides a powerful tool for combating piracy: finding the identities of compromised devices, revoking them and rendering the illegal boxes useless.

Caveat. The goal of a copyright protection mechanism is to create a legal channel of distribution of content and to disallow its abuse. As a consequence, an illegal distribution will require the establishment of alternative channels and should not be able to piggyback on the legitimate channel[2]. Such alternative

[1] For instance, the software clone known as DeCSS, that cracked the DVD Video "encryption", is shielded by a tamper-resistant software tool which makes it very hard to reverse engineer its code and know its details such as receivers identities or its decoding strategy.

[2] For instance in the case of cable TV the pirates should be forced to create their own cable network.

channels should be combated using other means and is not under the scope of the techniques developed in this paper, thought techniques such as revocation may be a useful deterrent against rogue users.

1.1 Related Work

Broadcast Encryption. The area of Broadcast Encryption was first formally studied (and the term coined) by Fiat and Naor in [24] 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.

To survey previous results we should fix our notation. An important parameter that is 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 [24] is one such scheme which allows the removal of any number of users as long as at most t of them collude. There the message length is $O(t \log^2 t)$, a user must store a number of keys that is logarithmic in t and the amount of work required by the user is $\tilde{O}(r/t)$ decryptions.

The logical-tree-hierarchy (LKH) scheme, suggested independently by Wallner et al. [48] and Wong et al. [49], is designed for the connected mode for multicast re-keying applications. It revokes a single user at a time, and updates the keys of all remaining users. If used in our scenario, it requires a transmission of $2r \log N$ keys to revoke r users, each user should store $\log N$ keys and the amount of work each user should do is $r \log N$ encryptions (the expected number is $O(r)$ for an average user). These bounds are somewhat improved in [12, 13, 34], but unless the storage at the user is extremely high they still require a transmission of length $\Omega(r \log N)$. The key assignment of this scheme and the key assignment of our first method are similar; see further discussion on comparing the two methods in Section 3.1.

Luby and Staddon [33] considered the information theoretic setting and devised bounds for any revocation algorithms under this setting. Their "Or Protocol" fits our Subset-Cover framework. We note in Section 3.3 that our second algorithm (the Subset Difference method), which is *not* information theoretic, beats their lower bound (Theorem 12 in [33]). Garay, Staddon and Wool [28] introduced the notion of *long-lived broadcast encryption*. In this scenario, keys of compromised decoders are no longer used for encryptions. The question they address is how to adapt the broadcast encryption scheme so as to maintain the security of the system for the good users.

The method of Kumar, Rajagopalan and Sahai [32] enables one-time revocation of up to r users with message lengths of $O(r \log N)$ and $O(r^2)$. Some of the improvements of our schemes as compared to those of [33, 32] stems from considering not the information theoretic setting but rather the computational one.

CPRM [17] is one of the methods that explicitly considers the stateless scenario. Our Subset Difference method outperforms CPRM by a factor of about 25 in the number of revocations it can handle when all the other parameters are fixed; Section 4.5 contains a detailed description and comparison.

Tracing Mechanisms. The notion of a tracing system was first introduced by Chor, Fiat and Naor in [14], and was later refined to the Threshold Traitor model in [37], [15]. Its goal is to 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 or clone using keys of at most t users. *Black-box tracing* assumes that only the outcome of the decoding box can be examined. [37], [15] provide combinatorial and probabilistic constructions that guarantee tracing with high probability. To trace t traitors, they require each user to store $O(t \log N)$ keys and to perform a single decryption operation. The message length is $4t$. The public key tracing scheme of Boneh and Franklin [7] provides a number-theoretic deterministic method for tracing. Note that in all of the above methods t is an *a-priori* bound.

Preventing Leakage of Keys. The problem of preventing illegal leakage of keys has been attacked by a number of quite different approaches. The legal approach, suggested by Pfitzmann [42], requires a method that not only traces the leaker but also yields a proof for the liability of the traitor (the user whose keys are used by an illegal decoder). Hence, leakage can be fought via legal channels by presenting this proof to a third party. The self enforcement approach, suggested by Dwork, Lotspiech and Naor [21], aims at *deterring* users from revealing their personal keys. The idea is to provide a user with personal keys that contain some sensitive information about the user which the user will be reluctant to disclose. The trace-and-revoke approach is to design a method that can trace the identity of the user whose key was leaked; in turn, this user's key is revoked from the system for future uses. The results in this paper fall into the latter category, albeit in a slightly relaxed manner. Although our methods assure that leaked keys will become useless in future transmissions, it may not reveal the actual *identities* of all leaking keys, thus somewhat lacking self-enforcement.

Content Tracing: In addition to tracing leakers who give away their private keys there are methods that attempt to detect illegal users who redistribute the content *after* it is decoded. This requires the assumption that good watermarking techniques with the following properties are available: it is possible to insert one of several types of watermarks into the content so that the adversary cannot create a "clean" version with no watermarks (or a watermark it did not receive). Typically, content is divided into segments that are watermarked separately. This setting with protection against collusions was first investigated by Boneh and Shaw [9]. A related setting with slightly stronger assumptions on the underlying watermarking technique was investigated in [25, 5, 44]. By introducing the *time* dimension, Fiat and Tassa [25] propose the dynamic tracing scenario in which the watermarking of a segment depends on feedback from the previous segment and which detects all traitors. Their algorithm was improved by Berkman, Parnas and Sgall [5], and a scheme which requires no real-time computation/feedback for this model was given in Safavani-Naini and Wang [44]. Content tracing is relevant to our scenario in that any content tracing mechanism can be combined with a key-revocation method to ensure that the traced users are indeed revoked and do not receive new content in the future. Moreover, the tracing methods of [25] are related to the tracing *algorithm* of Section 6.2.

Integration of tracing and revocation. Broadcast encryption can be combined with tracing schemes to yield trace-and-revoke schemes [3]. While Gafni et al. [27] and Stinson and Wei [46] only consider combinatorial constructions, the schemes in Naor and Pinkas [38] are computational constructions and hence more general. The previously best known trace-and-revoke algorithm of [38] 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 therefore these keys may be longer than symmetric keys. The tracing model of [38] is not a "pure" black-box model. Anzai et al. [2] employs a similar method for revocation, but without tracing capabilities. Our results improve upon this algorithm both in the work that must be performed at the user and in the lengths of the keys transmitted in the message.

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 [33]), called Subset-Cover algorithms. These algorithms are based on the princi-

[3] Note however that it is *not* the case that every system where users can be revoked and where tracing is possible is necessarily a trace-and-revoke scheme (see [15]).

ple 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[4]). Both instantiations are tree-based, namely the subsets are derived from a virtual tree structure imposed on all devices in the system[5]. 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 Section 3.3. 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 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 adapted 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.

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 Section 5). Section 3 describes two specific implementations of such algorithms. Section 4 presents extensions, implementation issues, public-key methods, application to multicasting as well as casting the recently proposed CPRM method (for DVD-audio and SD cards) in the Subset-Cover Framework. In Section 5 we define the "key-indistinguishability" property and provide the main theorem

[4] 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.

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

Method	Message Length	Storage @ Receiver	Processing time	no. 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

Figure 1: Performance Tradeoff for the Complete Subtree method and the Subset Difference method

characterizing the security of revocation algorithms in the Subset-Cover framework. Section 6 provides the traitors tracing extensions to Subset-Cover revocation algorithms and their seamless integration.

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 exact definition of the latter is provided in Section 5.

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[6] $S_{i_1}, \ldots, S_{i_m}$ 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 E_L 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.

[6] Whether the subsets in the cover are disjoint or not does not effect the revocation algorithm at all (may reduce the size), but is more important to the tracing. Even there it is possible to work with schemes that have overlapping subsets in the cover.

An exact discussion of security requirements of these primitives is given in Section 5. Some suggestions for the implementation of F_K and E_L are given in Section 4.1. 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:

1. Choose a session encryption key K.

2. Given a set $\mathcal{R}$ of revoked receivers, the center finds a partition of the users in $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets $S_{i_1}, \ldots, S_{i_m}$. Let $L_{i_1}, \ldots, L_{i_m}$ be the keys associated with the above subsets.

3. 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

$$\langle [i_1, i_2, \ldots, i_m, C_1, C_2, \ldots, C_m], M' \rangle :$$

1. Find i_j such that $u \in S_{i_j}$ (in case $u \in \mathcal{R}$ the result is **null**).

2. Extract the corresponding key L_{i_j} from I_u.

3. Compute $D_{L_{i_j}}(C_j))$ to obtain K.

4. Compute $D_K(M')$ to obtain and output M.

A particular implementation of such a 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_j and compute its key L_{S_j} from I_u. The algorithm is evaluated based upon three parameters:

i. 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 $\mathcal{N} \setminus \mathcal{R}$.

ii. 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.

iii. 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 [7].

2.3 Security of the Framework: Summary

The definition of the Subset-Cover framework allows a rigorous treatment of the security of any algorithm in this family, which is discussed in detail in Section 5. 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 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 Section 5 for details). We then proceed to show in Theorem 10 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.

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[8]. 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 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.

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

[8] Recently a method exhibiting various tradeoffs between the measures (bandwidth, storage and processing time) was proposed [36]. In particular it is possible to reduce the device storage down to $\log^2 n/\log D$ by increasing processing time to $D \log n$.

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. In other words, $u \in S_i$ iff v_i is an ancestor of u. The key assignment method is simple: assign 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 $u_1, \ldots, u_r$ be the leaves corresponding to the elements in $\mathcal{R}$. The method to partition $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets is as follows. Let $S_{i_1}, \ldots, S_{i_m}$ be all the subtrees of the original tree that "hang" off $ST(\mathcal{R})$, that is, all subtrees whose roots $v_1, \ldots, v_m$ 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: The Steiner tree $ST(\mathcal{R})$ has r leaves. An internal node is in $ST(\mathcal{R})$ iff it is on some path to a point in $\mathcal{R}$, therefore there are at most $r \log N$ nodes in $ST(\mathcal{R})$. A finer analysis of the number of modes in $ST(\mathcal{R})$ takes into account double counting of the nodes closer to the root and the fact that a node of outdegree 2 in $ST(\mathcal{R})$ does not produce a subset, and we can obtain the following:

Claim 1 *The number of subsets in a cover with N users and r revocations is at most $r \log(N/r)$.*

Proof: Note that the number of sets is exactly the number of degree 1 nodes in $ST(\mathcal{R})$. Assume by induction on the tree height that this is true for trees of depth i, i.e. that in a subtree with r leaves the maximum number of nodes of degree 1 is at most $r \cdot (i - \log r)$. Then consider a tree of depth $i+1$. If all the leaves are contained in one subtree of depth i, then by induction the total number of nodes of degree 1 is at most $r \cdot (i - \log r) + 1 \leq r \cdot (i + 1 - \log r)$. Otherwise, the number of nodes of degree 1 is the number of nodes of degree 1 in the left subtree (that has $r_1 \geq 1$ leaves) plus the number of nodes of degree 1 in the right subtree (that has $r_2 \geq 1$ leaves) and $r = r_1 + r_2$. By induction, this is at most $r_1 \cdot (i - \log r_1) + r_2 \cdot (i - \log r_2) = r \cdot i - (r_1 \log r_1 + r_2 \log r_2) \leq r \cdot (i + 1 - \log r)$ since $(r_1 \log r_1 + r_2 \log r_2) \geq r(\log r - 1)$. Note that this is also the average number of subsets (where the r leaves are chosen at random). $\Box$

The Decryption Step: Given a message

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

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.

There are several ways to facilitate an efficient search in this list[9]. First consider a generic method that works whenever each receiver is a member of relatively few subsets S_i: the values $i_1, i_2, \ldots i_m$ are put in a hash table and in addition a *perfect hash function* h of the list is transmitted as well (see [19] for a recent survey of such functions). The length of the description of h can be relatively small compared to the length of the list i.e. it can be $o(m \log w)$. The receiver u should check for all i such that $u \in S_i$ whether i is in the list by computing $h(i)$. In our case this would mean checking $\log N$ values.

Furthermore, suppose that we are interested in using as few bits as possible to represent the collection of subsets used $\{i_1, i_2, \ldots i_m\}$. The information-theoretic bound on the number of bits needed is $\lceil \log \binom{w}{m} \rceil$, which is roughly $m \log w/m$, using Stirling's approximation. (Note that when $m \approx \sqrt{w}$ this represents a factor 2 compression compared to storing $\{i_1, i_2, \ldots i_m\}$ explicitly.) However we are interested in a succinct representation of the collection that allows efficient lookup in this list. It turns out that with an *additive* factor

[9] This is relevant when the data is on a disk or buffered, rather than being broadcast, since broadcast results in scanning the list anyhow

of $O(m + \log\log w)$ bits it is possible to support an $O(1)$ lookup, see [10, 41]; the results they provide are even slightly better, but this bound is relatively simple to achieve.

It turns out that we can do even better for the complete subtree method, given the special structure of the subsets. For each node u, the desired ancestor i_j in the list is the one with which u and i_j have the longest common prefix. This means that a trie data structure with the revoked users allows finding it this prefix in time $\log N$ and by standard optimization this can be reduce to $\log r$ string comparisons, by balancing the $ST(\mathcal{R})$ tree via separators (nodes that split the tree into small connected components), as is done, for instance [35]. One can also guarantee a search using $\log\log N$ comparisons, by performing a binary search on this longest common prefix, where each prefix maintains it own collection of possible suffixes.

Summarizing, in the complete subtree method (i) the message header consists of at most $r \log \frac{N}{r}$ indices and encryptions of the session key (ii) receivers have to store $\log N$ keys and (iii) processing a message requires $O(\log\log N)$ operations plus a *single* decryption operation.

Security: 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 $i_1, i_2, \ldots i_m$, as stated above.

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 Definition 9.*

Comparison to the Logical Key Hierarchy (LKH) approach : Readers familiar with the LKH method of [48, 49] and the related Versakey framework [47] 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 Section 4.1.

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.

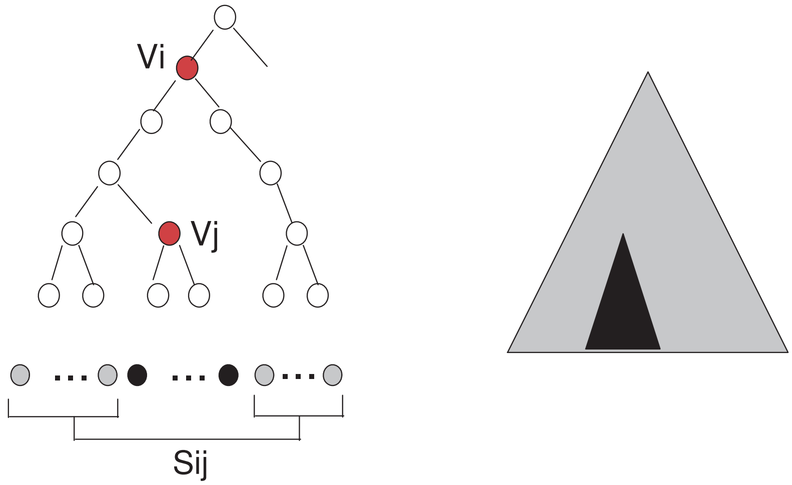

Figure 2: The Subset Difference Method: Subset $S_{i,j}$ contains all marked leaves (non-black).

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 given set $\mathcal{R}$ of revoked receivers, let $u_1, \ldots, u_r$ be the leaves corresponding to the elements in $\mathcal{R}$. The Cover is a collection of disjoint subsets $S_{i_1,j_1}, S_{i_2,j_2} \ldots, S_{i_m,j_m}$ which partitions $\mathcal{N} \setminus \mathcal{R}$. Below is an algorithm for finding the cover, and an analysis of its size (number of subsets).
Finding the Cover: The method partitions $\mathcal{N} \setminus \mathcal{R}$ into disjoint subsets $S_{i_1,j_1}, S_{i_2,j_2} \ldots, S_{i_m,j_m}$ as follows: let $ST(\mathcal{R})$ be the (directed) Steiner Tree induced by $\mathcal{R}$ and the root. We build 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 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

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.
The cover size: Lemma 3 shows that a cover can contain at most $2r-1$ subsets for any set of r revocations. Furthermore, if the set of revoked leaves is *random*, then the average number of subsets in a cover is $1.25r$.

Lemma 3 *Given any set of revoked leaves $\mathcal{R}$, the above method partitions $\mathcal{N} \setminus \mathcal{R}$ into at most $2r-1$ disjoint subsets.*

Proof: Every iteration increases the number of subsets by at most two (in step 2) and reduces the number of the Steiner leaves by one (in Step 3), except the last iteration that may not reduce the number of leaves but adds only one subset. Starting with r leaves, the process generates the total of $2r-1$ subsets. Moreover, every non-revoked u is in exactly one subset, the one defined by the first chain of nodes of outdegree 1 in $ST(R)$ that is encountered while moving from u towards the root. This encounter must hit a non-empty chain, since the path from u to the root cannot join $ST(R)$ in an outdegree 2 node, since this implies that $u \in \mathcal{R}$. □

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 Sections 4.2 and 6.2.

Lemma 4 *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.*

Proof: The proof is by induction on m. When $m = 1$, $\mathcal{S}$ contains a single set. Let this set be $S_{a,b}$, which is the set that is represented by two nodes in the tree (v_a, v_b). Denote by v_c and $v_{c'}$ the parent and the sibling of v_b respectively (it is possible that $v_a \equiv v_c$), and by r the root of the tree. Then the leaves in $\mathcal{N} \setminus \mathcal{U}$ are covered by the following two sets $S_{r,a}$ and $S_{c,c'}$. If $v_a \equiv v_c$ then the cover consists of a single set, $S_{r,c'}$.

To handle the case where $m > 1$, we need the following definition. We say that a set $S_{x,y}$ is *nested* within the set $S_{a,b}$ if the tree node v_x is contained in the subtree rooted at v_b. Note that if two subsets $S_{a,b}$ and $S_{a',b'}$ are disjoint but not nested, then the subtrees rooted at v_a and $v_{a'}$ must be disjoint[10]. Consider the following two cases:

1. All sets in $\mathcal{S}$ are maximal with respect to the nesting property. Let $S_{i_j} = S_{a_j,b_j}$ be the j^{th} set in $\mathcal{S}$. A cover for $\mathcal{N} \setminus \mathcal{U}$ is constructed by first covering all the subtrees rooted at the v_{b_j}'s, and then covering the rest of the leaves that are not contained in any one of the subtrees rooted at v_{a_j}. That is, for each

[10]The only exception is the case where b and b' are siblings and are both children of a. This is a degenerate case, and the two subsets should be replaced by a new subset consisting of the tree below a'

set S_{a_j,b_j} in $\mathcal{S}$, construct the set $S_{c,c'}$ where v_c and $v_{c'}$ are the parent and the sibling of v_{b_j} respectively for the total of m sets. To cover the rest, treat the nodes $v_{a_1}, \ldots, v_{a_m}$ as m revoked leaves and apply Lemma 3 to cover this tree. This requires $2m - 1$ additional sets, hence the number of sets required to cover $\mathcal{N} \setminus \mathcal{U}$ in this case is $3m - 1$.

2. $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ such that $|\mathcal{S}_1| = k \geq 1$ and there exists a maximal set $S_{a,b} \in \mathcal{S}_2$ with respect to the nesting property such that all sets in $\mathcal{S}_1$ are nested within $S_{a,b}$. Let $\mathcal{U}'$ be the subtree rooted at v_a. The idea is to first cover the leaves in $\mathcal{N} \setminus \mathcal{U}$ that are *not* in $\mathcal{U}'$ and then cover the ones in $\mathcal{N} \setminus \mathcal{U}$ that *are* in $\mathcal{U}'$. The first part of the cover can be obtained by applying the lemma recursively on the original tree with $\mathcal{S}_2$ where $S_{a,b}$ is replaced with the subset consisting of the tree below v_a. The second part is obtained by applying the lemma recursively on the tree rooted at v_b with $\mathcal{S}_1$. By the induction hypothesis, this requires the total number of $3(m-k) - 1 + 3k - 1 = 3m - 2$ sets.

□

Average-case analysis: The analysis of Lemma 3 is a worst-case analysis and there are instances which actually require $2r - 1$ sets. However, it is a bit pessimistic in the sense that it ignores the fact that a chain of nodes of outdegree 1 in $ST(\mathcal{R})$ may consist only of the end point, in which case no subset is generated. This corresponds to the case where $v_l \equiv v_i$ or $v_r \equiv v_j$ in Step 2. Suppose that the revoked set $\mathcal{R}$ is selected at random from all subsets of cardinality r of $\mathcal{N}$, then what is the expected number of subsets generated? The question is how many outdegree 1 chains are empty (i.e. contain only one point). We can bound it *from above* as follows: consider any chain for which it is known that there are k members beneath it. Then the probability that the chain is *not* empty is at most $2^{-(k-1)}$. For any $1 \leq k \leq r$ there can be at most r/k chains such that there are k leaves beneath it, since no such chain can be ancestor of another chain with k descendants. Therefore the expected number of non-empty chains is bounded by

$$\sum_{k=1}^{r} \frac{r}{k} \cdot \frac{1}{2^{k-1}} \leq 2r \sum_{k=1}^{\infty} \frac{1}{k} \cdot \frac{1}{2^k} \leq 2 \ln 2 \cdot r \approx 1.38 \cdot r.$$

Simulation experiments have shown a tighter bound of $1.25r$ for the random case. So the actual number of subsets used by the Subset Difference scheme is expected to be slightly lower than the $2r - 1$ worst case result.

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 subset 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 [29] to construct pseudo-random functions, which was also used by Fiat and Naor [24] as well as Canetti et al. [12] for purposes similar to ours.

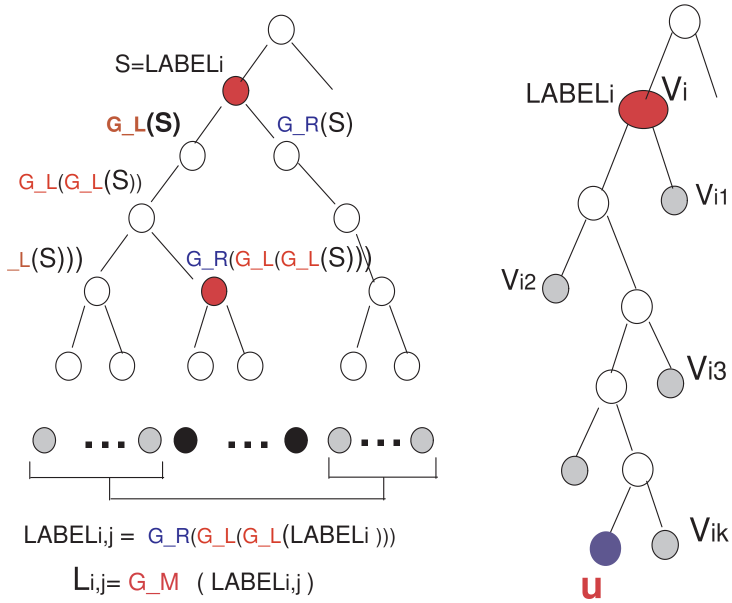

Figure 3: Key Assignment in the Subset Difference Method. *Left:* generation of $\mathrm{LABEL}_{i,j}$ and the key $L_{i,j}$. *Right:* leaf u receives the labels of $v_{i_1}, \ldots v_{i_k}$ that are induced by the label LABEL_i of v_i.

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 complete subtree T_i (rooted at v_i). We 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 Figure 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. More precisely, for a node v_j in tree T_i:

- Given the labels of all nodes except those that are v_j's ancestors *or* its descendants, the label $\mathrm{LABEL}_{i,j}$ is indistinguishable from random.
- Given the labels of all nodes except v_j the *key* $L_{i,j}$ is pseudo-random, but the label $\mathrm{LABEL}_{i,j}$, is *not* pseudo-random, simply because one can check for consistency with the labels of j's descendants.

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 allowing it to derive the key assignment described above. For each complete 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 Figure 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.

As for the total number of keys (in fact, labels) stored by receiver u, each complete subtree 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

$$1 + \sum_{k=1}^{\log N+1} k - 1 = 1 + \frac{(\log N + 1)\log N}{2} = \frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$$

Decryption Step: 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 outlined in Section 3.1 for table lookup structure, this subset can be found by $O(\min \log r, \log\log N)$ (though this is an overkill for the given method, since the other computation is much more expensive.) 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 (Definition 7 of Section 5) holds for this method, namely that each key is indistinguishable from a random key for all users not in the corresponding subset. Theorem 10 of Section 5 proves that this condition implies the security of the algorithm.

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 5 *The Subset Difference method satisfies the following properties: (i) The length of the message is at most $2r-1$ keys (ii) The storage at a receiver is $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$ keys and (iii) $O(\log N)$ operations plus a* single *decryption operation are required to decrypt a message. Moreover, the method is secure in the sense of Definition 9.*

3.3 Lower Bounds

Generic lower bound on message length

Any ciphertext in a revocation system when r users are revoked should clearly encode the original message plus the revoked subset, since it is possible to test which users decrypt correctly and which incorrectly using the preassigned secret information only (that was chosen independently of the transmitted message). Therefore we have a "generic" lower bound of $\log\binom{N}{r} \approx r\log N$ bits on the length of the header (or extra bits). Note that the subset difference method approaches this bound - the number of extra bits there is $O(r \cdot \text{ key-size})$.

3.3.1 Lower bounds for the information-theoretic case

If the keys to all the subsets are chosen independently (and hence u explicitly receives in I_u all L_i such that $u \in S_i$) then Luby and Staddon's lower bound for the "Or Protocol" [33] can be applied. They used the Sunflower Lemma (see below) to show that any scheme which employs m subsets to revoke r users must have at least one member with at least $\frac{\binom{N}{r}^{1/m}}{mr}$ keys. This means that if we want the number of subsets m to be at most r, then the receivers should store at least $\Omega(N/r^3)$ keys (as $\binom{N}{r} \geq \left(\frac{N}{r}\right)^r$). In the case where $r \ll N$, our (non-information-theoretic) Subset Difference method does better than this lower bound.

Note that when the number of subsets used in a broadcast is $O(r \log N)$ (as it is in the Complete Subtree method) then the above bound becomes useless. We now show that even if one is willing to use this many subsets (or even more), then at least $\Omega(\log N)$ keys should be stored by the receivers. We recall the Sunflower Lemma of Erdos and Rado (see [22]).

Definition 6 *Let $S_1, S_2, \ldots, S_\ell$ be subsets of some underlying finite ground set. We say that they are a* sunflower *if the intersections of any pair of the subsets are equal, in other words, for all $1 \leq i < j \leq \ell$ we have $S_i \cap S_j = \bigcap_{i=1}^{\ell} S_i$.*

The Sunflower Lemma says that in every set system there exists a sufficiently large sunflower: in a collection of N subsets each of size at most k there exists a sunflower consisting of at least $\frac{N^{1/k}}{k}$ subsets.

Consider now the sets $T_1, T_2, \ldots T_N$ of keys the receivers store. I.e. $T_u = \{L_i | u \in S_i\}$. If for all u we have that $|T_u| \leq k$, then there exists a sunflower of $\frac{N^{1/k}}{k}$ subsets. Pick one u such that T_u is in the sunflower and make $\mathcal{R} = \{u\}$. This means that in order to cover the other members of the sunflower we must use at least $\frac{N^{1/k}}{k} - 1$ keys, since no S_i can be used to cover two of the other members of the sunflower (otherwise S_i must also have the revoked u as a member). This means, for instance, that if $k = \sqrt{\log N}$ then just to revoke a single user requires using at least $\frac{2^{\sqrt{\log N}}}{\sqrt{\log N}} - 1$ subsets.

4 Further Discussions

4.1 Implementation Issues

Implementing E_L and F_K

One of the issues that arises in implementing a Subset-Cover scheme is how to implement the two cryptographic primitives E_L and F_K. The basic requirements from E_L and F_K were outlined above in Section 2. However, it is sometimes desirable to chose an encryption F that might be weaker (uses shorter keys) than the encryption chosen for E. The motivation for that is twofold: (1) to speed up the decoding process at the receiver (2) to shorten the length of the header. Such a strategy makes sense, for example, for copyright protection purposes. There it may not make sense to protect a *specific* ciphertext so that breaking it is very expensive; on the other hand we do want to protect the long lived keys of the system with a strong encryption scheme.

Suppose that F is implemented by using a stream cipher with a long key, but sending some of its bits in the clear; thus K corresponds to the hidden part of the key and this is the only part that needs to be encrypted in the header. (One reason to use F in such a mode rather than simply using a method designed with a small key is to prevent a preprocessing attack against F.) This in itself does not shorten the header, since it depends on the block-length of E (assuming E is implemented by block-cipher). We now provide a specification for using E, called Prefix-Truncation, which reduces the header length as well, in addition to achieving speedup, without sacrificing the security of the long-lived keys. Let $\text{Prefix}_i S$ denote the first

i bits of a string S. Let E_L be a block cipher and $\mathcal{U}$ be a random string whose length is the length of the block of E_L. 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 definition of E as described in Section 5. The Prefix-Truncated header is therefore:

$$\langle [\, i_1, i_2, \ldots, i_m, \mathcal{U}, [\text{Prefix}_{|K|}E_{L_{i_1}}(\mathcal{U})] \oplus K, \ldots, [\text{Prefix}_{|K|}E_{L_{i_m}}(\mathcal{U})] \oplus K \,], F_K(M) \rangle$$

Note that this reduces the length of the header down to about $m \times |K|$ bits long (say $56m$) instead of $m \times |L|$. In the case where the key length of E is marginal, then the following heuristic can be used to remove the factor m advantage that the adversary has in a brute-force attack which results from encrypting the same string $\mathcal{U}$ with m different keys. Instead, encrypt the string $\mathcal{U} \oplus i_j$, namely

$$\langle [\, i_1, i_2, \ldots, i_m, \mathcal{U}, [\text{Prefix}_{|K|}E_{L_{i_1}}(\mathcal{U} \oplus i_1)] \oplus K, \ldots, [\text{Prefix}_{|K|}E_{L_{i_m}}(\mathcal{U} \oplus i_m)] \oplus K \,], F_K(M) \rangle$$

All-Or-Nothing Encryptions for F_K

As before, we can imagine cases where the key used by F_K is only marginally long enough. Moreover, in a typical scenario like copyright protection, the message M is long (e.g. M may be a title on a CD or a DVD track). In such cases, it is possible to extract more security from the long message for a fixed number of key bits using the All-Or-Nothing encryption mode originally suggested by [43]. These techniques assure that the entire ciphertext must be decrypted before even a single message block can be determined. The concrete method of [43] results in a penalty of a factor of three in the numbers encryptions/decryptions required by a legitimate user; however, for a long message that is composed of l blocks, a brute-force attack requires a factor of l more time than a similar attack would require otherwise. Other All-Or-Nothing methods can be applied as well.

The drawback of using an All-Or-Nothing mode is its *latency*, namely the entire message M must be decoded before the first block of plaintext is known. This makes the technique unusable for applications that cannot tolerate such latency.

Frequently Refreshed Session Keys

Suppose that we want to prevent an illegal redistribution channel that uses some low bandwidth means, e.g. a low bandwidth label or a bootlegged CD, to send K, the session key. A natural approach to combat such channel is to encode different parts of the message M with different session keys, and to send all different session keys encrypted with all the subset keys. That is, send $l > 1$ different session keys all encrypted with the same cover, thus increasing the length of the header by a factor of l. This means that in order to have only a modest increase in the header information it is important that m, the number of subsets, will be as small as possible. Note that the number of decryptions that the receiver needs to perform in order to obtain its key $\mathcal{L}_{i_j}$ which is used in this cover remains one.

Storage at the Center

In both the Complete Subtree and Subset Difference methods, a unique label is associated with each node in the tree. Storing these labels explicitly at the Center can become a serious constraint. However, these labels can be generated at the center by applying a pseudo-random function on the name of the node without affecting the security of the scheme. This reduces the storage required by at the Center to the *single* key of the pseudo-random function.

Furthermore, it may be desirable to distribute the center between several servers with the objective of avoiding a single or few points of attack. In such a case the distributed pseudo-random functions of [39] may be used to define the labels.

Reducing Keys at the Receiver

A further optimization is a tradeoff between the number of labels at the receiver and the message length. One approach is to restrict the collection of subsets only to "shallow" subsets, namely sets $S_{i,j}$ such that v_i is at least at depth h from the root, where h is the tradeoff parameter. As a result, the cover size may increase by at most 2^h (additively), but the number of labels at the receiver is reduced to $\sum_{k=1}^{\log N+1-h} k-1 = \frac{1}{2}(\log N - h)^2 + \frac{1}{2}(\log N - h + 1)$.

4.2 Hierarchical Revocation

Suppose that the receivers are grouped in a hierarchical manner, and that it is desirable to revoke a group that consists of the subordinates of some entity, without paying a price proportional to the group size (for instance all the players of a certain manufacturer). Both methods of Section 3 lend themselves to hierarchical revocation naturally, given the tree structure. If the hierarchy corresponds to the tree employed by the methods, then to revoke the receivers below a certain node counts as just a single user revocation.

By applying Lemma 4 we get that in the Subset Difference Method we can remove any collection of m subsets and cover the rest with $3m-1$ subsets. Hence, the hierarchical revocation can be performed by first constructing m sets that cover all revoked devices, and then covering all the rest with $3m-1$, yielding the total of $4m$ sets.

4.3 Public Key methods

In some scenarios it is desirable to use a revocation scheme in a public-key mode, i.e. when the party that generates the ciphertext is not necessarily trustworthy and should not have access to the decryption keys of the users, or when ciphertexts may be generated by a number of parties. Any Subset-Cover revocation algorithm can be used in this mode: the Center (a trusted entity) generates the private-keys corresponding to the subsets and hands each user the private keys it needs for decryption. The (not necessarily trusted) party that generates the ciphertext should only have access to public-keys corresponding to the subsets which we call "the public-key file". That is, E is a public key cryptosystem whereas F is as before. In principle, any public key encryption scheme with sufficient security can be used for E. However, not all yield a system with a reasonable efficiency. Below we discuss the problems involved, and show that a Diffie-Hellman type scheme best serves this mode.

Public Key Generation: Recall that the Subtree Difference method requires that subset keys are derived from labels. If used in a public-key mode, the derivation yields random bits that are then used to generate the private/public key pair. For example, if RSA keys are used, then the random strings that are generated by the Pseudo Random Generator G can be used as the random bits which are input to the procedure which generates an RSA key. However, this is rather complicated, both in terms of the bits and time needed. Therefore, whenever the key assignment is not information-theoretic it is important to use a public-key scheme where the mapping from random bits to the keys is efficient. The Diffie-Hellman type scheme provides an efficient mapping.

Size of Public Key File: The problem is that the public key file might be large, proportional to w, the number of subsets. In the Complete Subtree method $w = 2N-1$ and in the Subtree Difference method it is $N \log N$. An interesting open problem is to come up with a public-key cryptosystem where it is possible to compress the public-keys to a more manageable size. For instance, an identity-based cryptosystem would be helpful for the information-theoretic case where keys are assigned independently. The recent proposals of Boneh and Franklin [8] and Cocks [16] fit this requirement.

Prefix-Truncated Headers: We would like to use the Prefix-Truncation, described in Section 4.1, with public-key cryptosystem to reduce the header size without sacrificing security of long-term keys. It can not be employed with an arbitrary public key cryptosystem (e.g. RSA). However, a Diffie-Hellman public key system which can be used for the Prefix-Truncation technique can be devised in the following manner. Interestingly, in such a system the length of public-key encryption is hardly longer than the private-key case.

Let G be a group with a generator g and let the subset keys be $L_1 = y_1, L_2 = y_2, \ldots, L_w = y_w$ elements in G. Let $g^{y_1}, g^{y_2}, \ldots, g^{y_w}$ be their corresponding public keys. Define h as a pairwise-independent function $h : G \mapsto \{0,1\}^{|K|}$ that maps elements which are randomly distributed over G to randomly distributed strings of the desired length (see e.g. [40] for a discussion of such functions). Given the subsets $S_{i_1}, \ldots, S_{i_m}$ to be used in the header, the encryption E can be 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

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

Interestingly, 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[11].

4.4 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 [49, 48]. 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. If 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 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 [49, 48] 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

[11] Both the scheme of Cramer and Shoup [18] and the random oracle based scheme [26] require some specific information for each recipient; a possible approach with random oracles is to follow the lines of [45].

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.

4.5 Comparison to CPRM

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 [17]. 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 $0 \le i \le A-1$ and $1 \le j \le C$ 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[12].

The CPRM method is a Subset-Cover method with two exceptions: (1) the subsets in a cover are not necessarily disjoint (which is not a problem) 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 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

[12] This is similar to the scenario considered in [28]

hierarchical revocation can be done very effectively, a property that the current CPRM algorithm does not have.

5 Security of the Framework

In this section we discuss the security of a Subset-Cover algorithm. Intuitively, we identify a critical property that is required from the key-assignment method in order to provide a secure Subset-Cover algorithm. We say that 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. We then proceed to show that any subset-cover algorithm that satisfies the key-indistinguishability property provides a secure encryption of the message.

We must specify what is a secure revocation scheme, i.e. describe the adversary's power and what is considered a successful break. We provide a sufficient condition for a Subset-Cover revocation scheme $\mathcal{A}$ to be secure. We start by stating the assumptions on the security of the encryption schemes E and F. All security definitions given below refer to an adversary whose challenge is of the form: distinguish between two cases (i) 'real" and (ii) 'random'.

5.1 Assumptions on the Primitives

Recall that the scheme employs two cryptographic primitives F_K and E_L. The security requirements of these two methods are different, since F_K uses short lived keys whereas E_L uses long-lived ones. In both cases we phrase the requirements in terms of a the probability of success in distinguishing an encryption of the true message from an encryption of a random message. It is well known that such formulation is equivalent to semantic security (that anything that can be computed about the message given the ciphertext is computable without it), see [30, 29, 4][13].

The method F_K for encrypting the body of the message should obey the following property: consider any feasible adversary $\mathcal{B}$ that chooses a message M and receives one of the following, where $K \in \{0,1\}^\ell$ is chosen uniformly at random:

i. $F_K(M)$, or the 'real' case.

ii. $F_K(R_M)$ for a random message R_M of length $|M|$, or the 'random' case.

The probability that $\mathcal{B}$ distinguishes the two cases is negligible and we denote the bound by ε_1, i.e.

$$|\Pr[\mathcal{B} \text{ outputs 'real' } |F_K(M)] - \Pr[\mathcal{B} \text{ outputs 'real' } |F_K(R_M)]| \leq \varepsilon_1. \tag{1}$$

Note that implementing F_K by a pseudo-random generator (stream-cipher) where K acts as the seed and whose output is Xored bit-by bit with the message satisfies this security requirement.

The long term encryption method E_L should withstand a more severe attack, in the following sense: consider any feasible adversary $\mathcal{B}$ that for a random key L gets to adaptively choose polynomially many inputs and examine E_L's encryption and similarly provide ciphertexts and examine E_L's decryption. Then $\mathcal{B}$ is faced with the following challenge: for a random plaintext x (which is provided in the clear) it receives one of the following

i. $E_L(x)$, or the 'real' case.

ii. $E_L(R_x)$ where R_x is a random string of length $|x|$, or the 'random' case.

[13] One actually has to repeat such an equivalence proof for the adversaries in question.

The probability that $\mathcal{B}$ distinguishes the two cases is negligible and we denote the bound by ε_2, i.e.

$$|\Pr[\mathcal{B} \text{ outputs 'real' } |E_L(x)] - \Pr[\mathcal{B} \text{ outputs 'real'}|E_L(R_x)]| \leq \varepsilon_2. \quad (2)$$

Note that the above specification indicates that E should withstand a chosen-ciphertext attack in the preprocessing mode in the terminology of [20] or CCA-I in [3]. Possible implementation of E_L can be done via pseudo-random permutations (which model block-ciphers). See more details on the efficient implementation of F and E in Section 4.1.

Key Assignment: Another critical cryptographic operation performed in the system is the key assignment method, i.e. how a user u derives the keys L_i for the sets S_i such that $u \in S_i$. We now identify an important property the key assignment method in a subset-cover algorithm should possess that will turn out to be sufficient to provide security for the scheme:

Definition 7 *Let $\mathcal{A}$ be a Subset-Cover revocation algorithm that defines a collection of subsets $S_1, \ldots, S_w$. Consider a feasible adversary $\mathcal{B}$ that*

1. *Selects i, $1 \leq i \leq w$*
2. *Receives the I_u's (secret information that u receives) for all $u \in \mathcal{N} \setminus S_i$*

We say that $\mathcal{A}$ satisfies the key-indistinguishability property *if the probability that $\mathcal{B}$ distinguishes L_i (the 'real' case) from a random key R_{L_i} of similar length (the 'random' case) is negligible and we denote this by ε_3, i.e.*

$$|\Pr[\mathcal{B} \text{ outputs 'real' } |L_i] - \Pr[\mathcal{B} \text{ outputs 'real' } |R_{L_i}]| \leq \varepsilon_3. \quad (3)$$

Note that all "information theoretic" key assignment schemes, namely schemes in which the keys to all the subsets are chosen independently, satisfy Definition 7 with $\varepsilon_3 = 0$.

The next lemma is a consequence of the *key-indistinguishability* property and will be used in the proof of Theorem 10, the Security Theorem.

Lemma 8 *For any $1 \leq i \leq w$ let $S_{i_1}, S_{i_2} \ldots, S_{i_t}$ be all the (distinct) subsets in the collection that are contained in S_i; Let $L_{i_1}, \ldots, L_{i_t}$ be their corresponding keys. For any adversary $\mathcal{B}$ that selects i, $1 \leq i \leq w$, and receives I_u for all $u \in \mathcal{N} \setminus S_i$, if $\mathcal{B}$ attempts to distinguish the keys $L_{i_1}, \ldots, L_{i_t}$ from random keys $R_{L_{i_1}}, \ldots, R_{L_{i_t}}$ (of similar lengths) then*

$$\left|\Pr[\mathcal{B} \text{ outputs 'real' } |L_{i_1}, \ldots, L_{i_t}] - \Pr[\mathcal{B} \text{ outputs 'real' } |R_{L_{i_1}}, \ldots, R_{L_{i_t}}]\right| \leq t \cdot \varepsilon_3.$$

Proof: Let us rename the subsets $S_{i_1}, S_{i_2} \ldots, S_{i_t}$ as $S_1, S_2, \ldots S_t$ and order them according to their size; that is for all $j = 1, \ldots, t, S_j \subseteq S_i$ and $|S_1| \geq |S_2| \geq \ldots |S_t|$. We will now use a hybrid argument: consider an "input of the j^{th} type" as one where the first j keys are the true keys and the remaining $t - j$ keys are random keys. $\forall 1 \leq j \leq t$, let p_j be the probability that $\mathcal{B}$ outputs 'real' when challenged with an input of the j^{th} type, namely

$$p_j = \Pr[\mathcal{B} \text{ outputs 'real' } |L_1, \ldots, L_j, R_{L_{j+1}}, \ldots, R_{L_t}]$$

Suppose that the lemma doesn't hold, that is $|p_t - p_0| > t \cdot \varepsilon_3$. Hence there must be some j for which $|p_j - p_{j-1}| > \varepsilon_3$. We now show how to create an adversary $\mathcal{B}'$ that can distinguish between R_{L_j} and L_j with probability $> \varepsilon_3$, contradicting the key-indistinguishability property. The actions of $\mathcal{B}'$ result from a simulation of $\mathcal{B}$:

- When $\mathcal{B}$ selects S_i, $\mathcal{B}'$ selects the subset $S_j \subseteq S_i$ from the above discussion (that is, the j for which $|p_j - p_{j-1}| > \varepsilon_3$). It receives I_u for all $u \in \mathcal{N} \setminus S_j$ and hence can provide $\mathcal{B}$ with I_u for all $u \in \mathcal{N} \setminus S_i$.

- When $\mathcal{B}'$ is given a challenge X and needs to distinguish whether X is R_{L_j} or L_j, it creates a challenge to $\mathcal{B}$ that will be $L_1, \ldots, L_j, R_{L_{j+1}}, \ldots, R_{L_t}$ or $L_1, \ldots, L_{j-1}, R_{L_j}, R_{L_{j+1}}, \ldots, R_{L_t}$. Note that due to their order (and distinctness) $S_1, \ldots, S_{j-1} \not\subseteq S_j$; since $\mathcal{B}'$ received I_u for all $u \in \mathcal{N} \setminus S_j$ it knows the keys $L_1, \ldots, L_{j-1}$, while $R_{L_{j+1}}, \ldots, R_{L_t}$ are chosen at random. The jth string in the challenge is simply X (the one $\mathcal{B}'$ received as a challenge.) $\mathcal{B}'$ response is simply $\mathcal{B}$'s answer to the query.

The advantage that $\mathcal{B}'$ has in distinguishing between R_{L_j} and L_j is exactly the advantage B' has in distinguishing between $L_1, \ldots, L_j, R_{L_{j+1}}, \ldots, R_{L_t}$ and $L_1, \ldots, L_{j-1}, R_{L_j}, R_{L_{j+1}}, \ldots, R_{L_t}$, which is by assumption larger than ε_3, contradicting the key-indistinguishability property. □

5.2 Security Definition of a Revocation Scheme

To define the security of a revocation scheme we first have to consider the power of the adversary in this scenario (and make pessimistic assumption on its ability). The adversary can pool the secret information of several users, and it may have some influence on the the 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 (again this is equivalent to semantic security).

Definition 9 *consider an adversary $\mathcal{B}$ that gets to*

1. *Select adaptively a set $\mathcal{R}$ of receivers and obtain I_u for all $u \in \mathcal{R}$. By adaptively we mean that $\mathcal{B}$ may select messages $M_1, M_2 \ldots$ and revocation set $\mathcal{R}_1, \mathcal{R}_2, \ldots$ (the revocation sets need not correspond to the actual corrupted users) and see the encryption of M_i when the revoked set is $\mathcal{R}_i$. Also $\mathcal{B}$ can create a ciphertext and see how any (non-corrupted) user decrypts it. It then asks to corrupt a receiver u and obtains I_u. This step is repeated $|\mathcal{R}|$ times (for any $u \in \mathcal{R}$).*

2. *Choose a message M as the challenge plaintext and a set $\mathcal{R}$ of revoked users that must include all the ones it corrupted (but may contain more).*

$\mathcal{B}$ then receives an encrypted message M' with a revoked set $\mathcal{R}$. It has to guess whether $M' = M$ or $M' = R_M$ where R_M is a random message of similar length. We say that a revocation scheme is secure if, for any (probabilistic polynomial time) adversary $\mathcal{B}$ as above, the probability that $\mathcal{B}$ distinguishes between the two cases is negligible.

5.3 The Security Theorem

We now state and prove the main security theorem, showing that the key-indistinguishability property is sufficient for a scheme in the subset-cover framework to be secure in the sense of Definition 9. Precisely,

Theorem 10 *Let $\mathcal{A}$ be a Subset-Cover revocation algorithm where the key assignment satisfies Definition 7 (the key-indistinguishability property) and where E and F satisfy the above requirements. Then $\mathcal{A}$ is secure in the sense of Definition 9 with security parameter $\delta \leq \varepsilon_1 + 2mw(\varepsilon_2 + 4w\varepsilon_3)$, where w is the total number of subsets in the scheme and m is the maximum size of a cover.*

Proof: Let $\mathcal{A}$ be a Subset-Cover revocation algorithm with the key indistinguishability property. Let $\mathcal{B}$ be an adversary that behaves according to Definition 9, where δ is the probability that $\mathcal{B}$ distinguishes between an encryption of M and an encryption of a random message of similar length.

Recall from Definition 9 that the adversary adaptively selects a set of receivers $\mathcal{R}$ and obtains I_u for all $u \in \mathcal{R}$. $\mathcal{B}$ then selects a challenge message M. Let $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$ be the cover of $\mathcal{N} \setminus \mathcal{R}$ defined by $\mathcal{A}$. As a challenge, $\mathcal{B}$ then receives an encrypted message and is asked to guess whether it encrypts M or a random message R_M of the same length as M. We consider $\mathcal{B}$'s behavior in case not all the encryptions are proper. Let a "ciphertext of the j^{th} type" be one where the first j subsets are noisy and the remaining subsets encode the correct key. In other words the body is the encryption using F_K and the header is:

$$[i_1, i_2, \ldots, i_m, E_{L_{i_1}}(R_K^1), E_{L_{i_2}}(R_K^2), \ldots, E_{L_{i_j}}(R_K^j), E_{L_{i_{j+1}}}(K), \ldots, E_{L_{i_m}}(K)]$$

where K is a random key and $\{R_K^i\}$ are random strings of the same length as the key K. Let Δ_j be the advantage that for a ciphertext of the j^{th} type $\mathcal{B}$ distinguishes between the cases where $F_K(M)$ or $F_K(R_M)$ are the body of the message. I.e.

$$\Delta_j = |\Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(M)] - \Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(R_M)]|\,,$$

where the header is of the j^{th} type.

The assumption that $\mathcal{B}$ can break the revocation system implies that $\Delta_0 = \delta$. We also know that $\Delta_m \leq \varepsilon_1$, the upper bound on the probability of breaking F_K as defined by (1), since in ciphertexts of the m^{th} type the encryptions $E_{L_{i_j}}$ in the header contain no information on the key K used for the body so K looks random to $\mathcal{B}$. Hence there must be some $0 < j \leq m$ such that

$$|\Delta_{j-1} - \Delta_j| \geq \frac{\delta - \varepsilon_1}{m}.$$

We now apply the inequality $|a-b| - |c-d| \leq |a-c| + |b-d|$ and conclude that for this j it must be the case that for either M or R_M the difference in the probability that $\mathcal{B}$ outputs 'real' between the case when the header is of the j^{th} type and when it is of the $(j-1)^{th}$ type (and the same message is in the body) is at least $\frac{\delta - \varepsilon_1}{2m}$. In other words either

$$\begin{aligned} | \quad & \Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(M) \text{ header is of } j^{th} \text{ type}] \\ - \quad & \Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(M) \text{ header is of } (j-1)^{th} \text{ type}] \,| \geq \frac{\delta - \varepsilon_1}{2m} \end{aligned} \tag{4}$$

or

$$\begin{aligned} | \quad & \Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(R_M) \text{ header is of } j^{th} \text{ type}] \\ - \quad & \Pr[\mathcal{B} \text{ outputs 'real' } | \text{body is } F_K(R_M) \text{ header is of } (j-1)^{th} \text{ type}] \,| \geq \frac{\delta - \varepsilon_1}{2m} \end{aligned} \tag{5}$$

The intuition for proceeding is that a ciphertext of the $(j-1)^{th}$ type is noticeably different from a ciphertext of the j^{th} type only if it is possible to distinguish between $E_{L_{i_j}}(K)$ and $E_{L_{i_j}}(R_K)$. Therefore, the change in the distinguishing advantage $|\Delta_{j-1} - \Delta_j| \geq \frac{\delta - \varepsilon_1}{m}$ can be used to either break the encryption E_L or to achieve an advantage in distinguishing the keys. We now show how $\mathcal{B}$ can be used to construct an adversary $\mathcal{B}'$ that either breaks E_L or breaks the key-indistinguishability property, as extended by Lemma 8. This in turn is used to derive bounds on δ:

- $\mathcal{B}'$ picks at random $1 \leq i \leq w$ and asks to obtain I_u for all $u \notin S_i$; this is a guess that $S_{i_j} = S_i$. Let $S_{k_1}, S_{k_2}, \ldots, S_{k_t}$ be the subsets of the collection contained in S_i.

- $\mathcal{B}'$ receives either $L_0, L_{k_1}, \ldots, L_{k_t}$ or $R_{L_0}, R_{L_{k_1}}, \ldots, R_{L_{k_t}}$ where $L_0 = L_i$, the key of the subset S_i, and $L_0, L_{k_1}, \ldots, L_{k_t}$ are the keys of $S_{k_1}, S_{k_2}, \ldots, S_{k_t}$, as in Lemma 8. It attempts to distinguish between the case where the input corresponds to true keys and the case where the input consists of random keys.
- $\mathcal{B}'$ simulates $\mathcal{B}$ as well as the Center that generates the ciphertexts and uses $\mathcal{B}$'s output:
  - When the Center is faced with the need to encrypt (or decrypt) using the key of subset S_j such that $S_j \not\subseteq S_i$, then it knows at least one $u \in S_j$; from I_u it is possible to obtain L_j and encrypt appropriately. If $S_j \subseteq S_i$ then $S_j = S_{k_h}$ for some $1 \leq h \leq t$ and $\mathcal{B}'$ uses the key that was provided to it (either L_{k_h} or $R_{L_{k_h}}$).
  - When $\mathcal{B}$ decides to corrupt a user u, if $u \notin S_i$, then $\mathcal{B}'$ can provide it with I_u. If $u \in S_i$ then the guess that $i_j = i$ was wrong and we abort the simulation.
  - When the Center needs to generate the challenge ciphertext M for $\mathcal{B}$, $\mathcal{B}'$ finds a cover for $\mathcal{R}$, the set of users corrupted by $\mathcal{B}$. If $i_j \neq i$, then the guess was wrong and the simulation is aborted. Otherwise a random key K is chosen and a body of a message encrypted with K is generated where the encrypted message is either M or R_M (depending to whom the difference between Δ_j and Δ_{j-1} is due, i.e. whether (4) or (5) holds) and one of two experiments is performed:

    Experiment j: Create a header of ciphertext of the j^{th} type.

    Experiment $j-1$: Create a header of ciphertext of the $(j-1)^{th}$ type.

    Provide as challenge to $\mathcal{B}$ the created header and body.
- If the simulation was aborted, choose to output 'real' or 'random' uniformly at random. Otherwise provide $\mathcal{B}$'s output.

Denote by P_L^j (and P_L^{j-1} resp.) the probability that in experiment j (resp. $j-1$) in case the input to $\mathcal{B}'$ are the true keys the simulated $\mathcal{B}$ outputs 'real'; denote by P_R^j (and P_R^{j-1} resp.) the probability that in experiment j (experiment $j-1$) in case the input to $\mathcal{B}'$ are random keys the simulated $\mathcal{B}$ outputs 'real'. We claim that the differences between all these 4 probabilities can be bounded:

Claim 11 $|P_L^j - P_L^{j-1}| \geq \frac{\delta-\varepsilon_1}{2wm}$

Proof: In case the input to $\mathcal{B}'$ are the true keys, the resulting distribution that the simulated $\mathcal{B}$ experiences is what $\mathcal{B}$ would experience in a true execution (where the difference between a j^{th} ciphertext and $(j-1)^{\text{th}}$ ciphertext are at least $\frac{\delta-\varepsilon_1}{2m}$). The probability that the guess was correct is $1/w$ and this is independent of the action of $\mathcal{B}$, so we can experience a difference of at least $\frac{\delta-\varepsilon_1}{2wm}$ between the two cases. □

Claim 12 $|P_R^j - P_R^{j-1}| \leq \varepsilon_2$ *where* ε_2 *is defined by (2).*

Proof: Since otherwise we can use $\mathcal{B}'$ to attack E: whenever there is a need to use the key corresponding to the set S_i, ask for an encryption using the random key. Similarly use the K in the challenge for E as the one in the challenge of $\mathcal{B}'$. □

Claim 13 $|P_R^j - P_L^j| \leq w \cdot \varepsilon_3$ *and* $|P_R^{j-1} - P_L^{j-1}| \leq w \cdot \varepsilon_3$*, where* ε_2 *is defined by (3).*

Proof: If any of the two inequalities does not hold, then we can use $\mathcal{B}'$ as an adversary for Lemma 8 and contradict the safety of the key assignment (we know that $t \leq w$). □

From these three claims and applying the inequality $|a-b| - |c-d| \leq |a-c| + |b-d|$ we can conclude that

$$\frac{\delta - \varepsilon_1}{2wm} - \varepsilon_2 \leq 2w \cdot \varepsilon_3$$

and hence the overall security parameter of $\mathcal{A}$ satisfies $\delta \leq \varepsilon_1 + 2mw(\varepsilon_2 + 2w\varepsilon_3)$.

□

Weaker notions of security It is interesting to deal with the case where the encryption provided by F is not so strong. To combat copyright piracy it may not make sense to protect a *specific* ciphertext so that breaking it is very expensive; on the other hand we do want to protect the long lived keys of the system. The security definition (Definition 9) can easily be adapted to the case where distinguishing $F_K(M)$ from $F_K(R_M)$ cannot be done in some time T_1 where T_1 is not too large (this may correspond to using a not very long key K): the challenge following the attack is to distinguish $F_K(M)$ from $F_K(R_M)$ it time less than T_1' not much smaller than T_1. Essentially the same statement and proof of security as Theorem 10 hold. The fact that retrieving K does not have to be intractable, just simply expensive, means that K does not necessarily have to be long; see discussion on the implications on the total message length in Section 4.1.

It is also possible to model the case where the protection that F_K provides is not indistinguishability (e.g. F_K encrypts only parts of the message M that are deemed more important). In this case we should argue that the header does not provide more information regarding M than does $F_K(M)$. More precisely, suppose that $\mathcal{M}$ is a distribution on messages M and let $\mathcal{B}$ be an adversary that attacks the system as in Definition 9 but is given as a challenge a valid encryption of a message $M \in_R \mathcal{M}$ and attempts to compute some function of M (e.g. M defines a piece of music and the function is to map it to sounds). A scheme is considered secure if for any $\mathcal{M}$ and $\mathcal{B}$ there is a $\mathcal{B}'$ that simply receives $F_K(M)$ without the header and (i) performs an amount of work proportional to $\mathcal{B}$ after receiving the challenge and (ii) whose output is indistinguishable from $\mathcal{B}$'s output; the distinguisher should have access to M. Here again for any subset cover algorithm where E and the key assignment algorithm satisfy the requirements of Section 5.1 the resulting scheme will satisfy the relaxed definition.

6 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 (or more than one box) 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 [37] (which requires an identification of the traitor's identity) and in particular it lacks *self enforcement* [21]. 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 reset 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 $\mathcal{S}$ into subsets that renders the box useless, i.e. given an encryption with the partition $\mathcal{S}$ the box decrypts it correctly 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 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) The size of the coalitions against whom it is resilient (iii) The number of queries needed to perform the tracing. Our algorithm is resilient to any number of traitors and does not require any changes to the subset cover algorithm, provided it has the bifurcation property.

6.1 The Tracing Algorithm

Subset tracing: An important procedure in our tracing mechanism is one that when given a partition $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$ and an illegal box outputs one of two possible outputs: either (i) 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. We describe it in detail in Section 6.1.1.

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) disjoint and roughly equal in size subsets, i.e. that there exists $1 \leq i_1, i_2 \leq w$ such that $S_i = S_{i_1} \cup S_{i_2}$, $S_{i_1} \cap S_{i_2} = \phi$ 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 compared with the size of the original 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 Figure 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 $S_{i_1}, S_{i_2}, \ldots S_{i_m}$ of disjoint subsets. 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 $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$. 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 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.

Analysis: Since a partition can occur only in a subset that has a traitor and contains more than one element and since the cover consists of disjoint subsets, it follows that each traitor can force at most $\log_a N$, where a

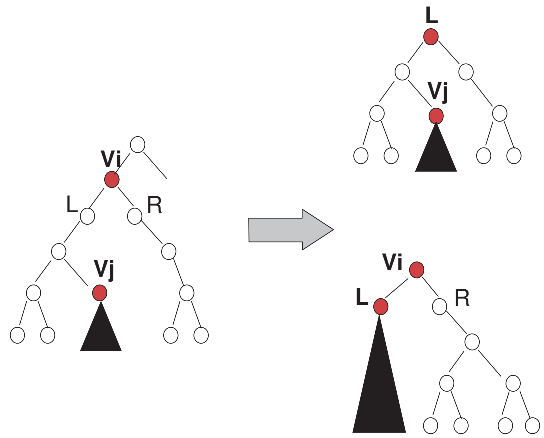

Figure 4: Bifurcating a Subset Difference set $S_{i,j}$, depicted in the left. The black triangle indicates the excluded subtree. L and R are the left and the right children of v_i. The resulting sets $S_{L,j}$ and $S_{i,L}$ are depicted to the right.

is the inverse of the bifurcation value. Altogether the number of iterations can be at most $t \log_a N$. 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 $\log_a N$.

6.1.1 The Subset Tracing Procedure

The Subset Tracing procedure first tests whether the box decodes correctly a message with the given partition $\mathcal{S} = S_{i_1}, S_{i_2}, \ldots S_{i_m}$, i.e. succeeds with probability p 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

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

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 $|p_{j-1} - p_j| \geq \frac{p}{m}$.

Intuitively, if there is a difference between p_j and p_{j-1} then either the adversary's coalition contains a member of S_{i_j} or the adversary has managed to break the encryption scheme E or the key assignment method. This is formalized in the next claim.

Claim 14 *Let ε be an upper bound on the sum of the probabilities of breaking the encryption scheme E and key assignment method, i.e. $\varepsilon = \varepsilon_2 + \varepsilon_3$, where ε_2 and ε_3 are defined by (2) and (3). Let γ be such that it is not possible to break E or the key assignment scheme in time $1/\gamma^2$. Consider the event A defined as "S_{i_j} does not contain a traitor and $|p_j - p_{j-1}| \geq \gamma$". Then $\Pr[A] \leq \varepsilon \cdot w \cdot m$*

Proof: This is essentially as the proof of Theorem 10. We can guess j and the set i_j and run a simulation knowing I_u for all $u \in \mathcal{N} \setminus S_{i_j}$. If we guessed correctly, which happens with probability at least $1/wm$, then this can be verified in time roughly $1/\gamma^2$. In this case we can either break the key assignment or E. $\Box$

Therefore if we set $\gamma = q/m$ we get that with at most negligible probability a set not containing a traitor is implicated. Note that the time and probability of breaking F does not appear.

We now describe a binary-search-like method that efficiently finds a pair of values (p_j, p_{j-1}) among $p_0, p_1, \ldots, p_m$ satisfying $|p_{j-1} - p_j| \geq \frac{p}{m}$ (or at least roughly so.) Starting with the entire interval $[1, m]$, the search is repeatedly narrowed down to an arbitrary interval $[a, b]$. At each stage, the middle value $p_{\frac{a+b}{2}}$ is computed (approximately) and the interval is further halved either to the left half or to the right half, depending on difference between $p_{\frac{a+b}{2}}$ and the endpoint values p_a and p_b of the interval and favoring the interval with the larger difference. The method is outlined below; it outputs the index j.

```
SubsetTracing(a, b, p_a, p_b)

If (a == b - 1)
   return b
Else
   c = ⌈(a+b)/2⌉
   Find p_c
   If |p_c - p_a| ≥ |p_b - p_a|
      SubsetTracing(a, c, p_a, p_c)
   Else
      SubsetTracing(c, b, p_c, p_b)
```

Let us first analyze this procedure under the idealized assumption that p_c is computed precisely. The above procedure takes at most $\lceil \log m \rceil$ steps before it gets to an interval of length 1. It reduces in each iteration the difference between the probabilities of the endpoints by at most a half yielding a difference of at least $(p_m - p_0)/2m$ at the end (the $2m$ factor is from the fact that $2^{\lceil \log m \rceil} \leq 2m$.)

However, we do not have a perfect mechanism for estimating p_c, so we have to deal with errors and mistakes. Suppose that we have a procedure that with probability at least $1 - \varepsilon$ can decide the following for some value δ:

1. If $\frac{|p_c - p_a|}{|p_b - p_a|} > \frac{1}{2}(1 + \delta)$, return "$|p_c - p_a| > |p_b - p_c|$"
2. If $\frac{|p_c - p_a|}{|p_b - p_a|} < \frac{1}{2}(1 - \delta)$, return "$|p_c - p_a| < |p_b - p_c|$
3. Otherwise, any decision is acceptable.

Since the estimation procedure is applied $\lceil \log m \rceil$ times in SubsetTracing, set $\delta = \frac{1}{\lceil \log m \rceil}$. At each step with probability at least $1 - \varepsilon$ the interval $|p_b - p_a|$ shrinks by at most a factor of $\frac{1}{2}(1 - \delta)$, so at the i^{th} step the difference between the endpoints is (with probability at least $1 - i \cdot \varepsilon$) larger than $(\frac{1}{2}(1 - \delta))^i$. When the search finishes we have that with probability at least $1 - \varepsilon \lceil \log m \rceil$ the difference between the endpoints is at least

$$\frac{p}{2m} \cdot \left(1 - \frac{1}{\lceil \log m \rceil}\right)^{\lceil \log m \rceil} \geq \frac{p}{2em} \stackrel{\text{def}}{=} \Delta.$$

Let X_c be a $\{0, 1\}$ random variable satisfying $\Pr[X_c = 1] = p_c$. We can easily sample such a random variable by checking whether the decryption is correct when the first c subsets are noisy. We assume that we have good estimates p'_a for p_a and p'_b for p_b from previous iterations and only need to estimate p'_c. In order to decide whether "$|p_c - p_a| > |p_b - p_c|$" or "$|p_c - p_a| < |p_b - p_c|$" apply the following procedure:

- Sample $n = \frac{\ln \frac{2}{\varepsilon}}{2(\frac{\Delta\delta}{4})^2}$ times from the distribution X_c and compute p'_c, the estimate for p_c, by taking it to be the number of 1's divided by n.

- If $p'_c > \frac{p'_a+p'_b}{2}$ then decide "$|p_c - p_a| > |p_b - p_c|$"
- If $p'_c < \frac{p'_a+p'_b}{2}$ then decide "$|p_c - p_a| < |p_b - p_c|$"

Claim 15 *If the following conditions hold (i) $p'_a \in (p_a - \frac{\Delta\delta}{4}, p_a + \frac{\Delta\delta}{4})$, (ii) $p'_b \in (p_b - \frac{\Delta\delta}{4}, p_b + \frac{\Delta\delta}{4})$ and (iii) $p'_c \in (p_c - \frac{\Delta\delta}{4}, p_c + \frac{\Delta\delta}{4})$ and $|p_a - p_b| \geq \Delta$, then a decision satisfying requirements 1, 2 and 3 above is made.*

Proof: If $\frac{|p_c-p_a|}{|p_b-p_a|} > \frac{1}{2}(1+\delta)$ then by substituting $\Delta \leq |p_b - p_a|$ we get that $p_c > \frac{p_b+p_a}{2} + \frac{\Delta\delta}{2}$. Combining this with the above, $p'_c \geq p_c - \frac{\Delta\delta}{4} > \frac{p_b+p_a}{2} + \frac{\Delta\delta}{4} \geq \frac{p'_a+p'_b}{2}$ so the correct decision is reached. Similarly, if $\frac{|p_c-p_a|}{|p_b-p_a|} < \frac{1}{2}(1-\delta)$. □

The next claim shows that assuming these conditions hold for p_a and p_b they also hold for p_c with high probability:

Claim 16 *If $n = -8\log\varepsilon/(\Delta^2\delta^2)$, then $p'_c \in (p_c - \frac{\Delta\delta}{4}, p_c + \frac{\Delta\delta}{4})$ with probability at least $1-\varepsilon$.*

Proof: We apply the variant of Chernoff bounds described in [1], Corollary A.7 [p. 236], stating that for a sequence of mutually independent random variables $Y_1, \ldots, Y_n$ satisfying $\Pr[Y_i = 1-p] = p$ and $\Pr[Y_i = -p] = 1-p$ (i.e. $E[Y_i] = 0$) it holds that

$$\Pr[|\sum_{i=1}^{n} Y_i| > t] < 2e^{-2t^2/n}$$

for any $t > 0$. When estimating p_c we have n random variables each of which is 1 with probability p_c and we take p'_c to be the relative ratio of 1's. Therefore we have that

$$\Pr[|p'_c - p_c| > \frac{t}{n}] = \Pr[|np'_c - np_c| > t] < 2e^{-2t^2/n}.$$

Hence, by choosing $n = \frac{-8\log\varepsilon}{\Delta^2\delta^2}$ and $t = n \cdot \frac{\Delta\delta}{4}$ we get that $\Pr[|p'_c - p_c| > \Delta\delta/4]$ is at most ε. □

It follows that a subset tracing procedure that works with success probability of $(1-\varepsilon\log m)$ requires at most $O(m^2 \log\frac{1}{\varepsilon}\log^3 m)$ ciphertext queries to the decoding box over the entire procedure. Note that a total probability of success bounded away from zero is acceptable, since it is possible to verify that the resulting p_{j-1}, p_j differ, and hence ϵ can be $O(1/\log m)$.

Noisy binary search: A more sophisticated procedure is to treat the Subset-Tracing procedure as *noisy binary search*, as in [23]. They showed that in a model where each answer is correct with some fixed probability (say greater than $2/3$) that is independent of history it is possible to perform a binary search in $O(\log N)$ queries. Each step might require backtracking; in the subset-tracing scenario, the procedure backtracks if the condition $|p_a - p_b| \geq (\frac{1}{2})^i$ does not hold at the i^{th} step (which indicates an error in an earlier decision). Estimating the probability values within an accuracy of $\frac{1}{m}$ while guaranteeing a constant probability of error requires only $O(m^2)$ ciphertexts queries. This effectively means that we can fix δ and ϵ to be constants (independent of m). Therefore, we can perform the noisy binary search procedure with $O(m^2\log m)$ queries.

6.2 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?

Suppose that there is a bound B on the length of the header (e.g. it is a hardware limitation in a disk.) Recall that the Complete Subtree method requires a message length of $r \log(N/r)$ for r revocations, hence roughly $B/\log N$ users can be revoked and this number can also be traced using the above algorithm. However, in the Subset Difference method, since the message length is at most $2r - 1$, $O(B)$ users can be revoked, but only $O(B/\log N)$ traitors can be traced by the above algorithm (no better than the Complete Subtree method.) This is the motivation for the *Frontier subsets* optimization.

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.[14] 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 $(S_{i_{j_1}}, S_{i_{j_2}})$ is said to be in the frontier if $S_{i_{j_1}}$ and $S_{i_{j_2}}$ resulted from a split-up of a single subset at an earlier iteration. Also neither ($S_{i_{j_1}}$ nor $S_{i_{j_2}}$) 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 $F \subset \mathcal{S}$ 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. If $|S_{i_j}| = 1$ then its member is a traitor and we permanently remove it. Otherwise, split S_{i_j} into $S_{i_{j_1}}$ and $S_{i_{j_2}}$.
- $F' = F \cup S_{i_{j_1}} \cup S_{i_{j_2}}$ ($S_{i_{j_1}}$ and $S_{i_{j_2}}$ 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'.

Why does the above process converge? The key point is that though the number of frontier sets does not necessarily increase, in each iteration the number of new *small* frontier sets always increases by at least one. More precisely, at the end of each iteration construct a vector V of length N where for all $1 \leq i \leq N$ the ith entry is the number of sets of size i in the frontier. Consider a lexicographic order of these vectors, i.e. one where the largest vector has the value N in entry 1. This vector V increases at each iteration:

Claim 17 *For every iteration where a traitor is not isolated the vector V increases lexicographically.*

Proof: Suppose that a set S_{i_j} of size ℓ has been found at by the Subset Tracing procedure. If this set had not been in the frontier, then clearly V only increases lexicographically. If S_{i_j} was in the frontier and its

[14]This idea is similar to the second scheme of [25], Section 3.3. However, in [25] the merge is straightforward as their model allows any subset. In our model only subsets from the predefined collection are allowed, hence a merge which produces subsets of this particular type is non-trivial.

buddy was of size ℓ', then from the bifurcation property it must be the case that $\ell' \geq 2\ell$. Let $|S_{i_{j_1}}| = \ell_1$ and $|S_{i_{j_2}}| = \ell_2$ and assume wlog that $\ell_1 \leq \ell_2$. We have that $\ell_1 \leq \ell/2 \leq \ell'$ and $\ell_2 < \ell$. Therefore in the vector V we have that entries ℓ and ℓ' lose one but two other entries, one no larger than ℓ' and the other one strictly smaller than ℓ gain one, so V is increased lexicographically. $\square$

Therefore the process must stop when or before all sets in the frontier are singletons. Note that the claim also implies that the total number of iterations is at most $t \log_{3/2} N$, since tracing back each singleton set to the time it entered the frontier involves at most $\log_{3/2} N$ iterations.

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 4 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.

6.3 Tracing Traitors from Many Boxes

As new illegal decoding boxes, decoding clones and hacked keys are continuously being introduced during the lifetime of the system, the trace and revoke strategy should be able to respond to the changes. This strategy is now as follows: at any point in time maintains a partition $\mathcal{S}$ that renders all the illegal boxes found so far invalid. When a new box is introduced, run the tracing algorithm *in parallel* on all boxes (the previous ones plus the new one) by providing the same input to all the boxes. When the *first* box detects a traitor in one of the sets we re-partition accordingly and the new partition is now input to *all* boxes simultaneously. Note that boxes from previous rounds where we had a partition that disabled them might suddenly resurrect (under a new partition) and start decrypting correctly. Hence the need to run the tracing algorithm on all boxes simultaneously. The output of this simultaneous algorithm is a partition (or "revocation strategy") that renders *all* revoked receivers and illegal black boxes invalid.

7 Further Work

This work raises several interesting directions, among them:

Public key file size: Is it possible to obtain a method corresponding to the subset difference for public-keys, but one where the size of the public key file is small. The difficulty is dealing with a private-key that is generated by GGM like labeling in a public-key setting.

Public key - prefix truncation: Is it possible to broadcast the same message to a large group of recipients with a small marginal increase in the size of the transmission per recipient while maintaining chosen ciphertext security and without resorting to random oracles.

Better Constructions: Is it possible to improve in terms of broadcast length and storage requirements on the methods presented in this paper. Recently, Halevy and Shamir [31] have shown a variant of Subset Difference called LSD (Layered Subset Difference), where each subset S_{ij} from the Subset Difference method is represented as a union of a few other subsets, forming a smaller collections altogether. The storage requirements are reduced to $O(\log^{1+\varepsilon} N)$ while the header length is $O(r/\varepsilon)$, providing a full spectrum between the Complete Subtree and Subset Difference methods. A reasonable choice is $\varepsilon = 2$. Are further generalizations possible?

Better Lower Bounds: Is is possible to obtain better (than those of Section 3.3.1) lower bounds for the information-theoretic case as well as deriving non-trivial lower bounds for the computational case.

Broadcast encryption: One of the scenarios this work does not resolve is when one performs a broadcast to a "medium" sized set, e.g. $\sqrt{N}$ of the user and one which does *not* have a hierarchical representation (recall that Section 4.2 deals with the latter.) In general, if we combine the trivial method of sending as separate encryption to each non revoked user with the subset difference method, then $O(\min\{N - r, r\})$ encryptions are sufficient. The question is whether one can do much better, in case the revoked set is known to the users.

Acknowledgements

We thank Omer Horvitz for many comments regarding the paper and the implementation of the system. We thank Ravi Kumar, Nelly Fazio and Florian Pestoni for useful comments and the anonymous referees for helpful suggestions.

References

[1] N. Alon and J. Spencer, **The Probabilistic Method**, John Wiley & Sons, 1992.

[2] J. Anzai, N. Matsuzaki and T. Matsumoto, *A Quick Group Key Distribution Sceheme with "Entity Revocation"*, Advances in Cryptology - Asiacrypt '99, Lecture Notes in Computer Science 1716, Springer, 1999, pp. 333–347.

[3] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway: *Relations Among Notions of Security for Public-Key Encryption Schemes*, Advances in Cryptology - CRYPTO'98, Lecture Notes in Computer Science 1462, Springer, 1998, pp. 26–45.

[4] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, *A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation,* Proc. of 38th IEEE Symposium on Foundations of Computer Science, 1997, pp. 394–403.

[5] 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.

[6] M. Blum and S. Micali, How to Generate Cryptographically Strong Sequences of Pseudo Random Bits, SIAM J. Compt., Vol. 13, pp. 850–864, 1984.

[7] D. Boneh and M. Franklin, An efficient public key traitor tracing scheme. Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag, pp. 338–353, 1999.

[8] D. Boneh and M. Franklin, Identity Based Encryption from the Weil Pairing, Advances in Cryptology - CRYPTO 2001, Lecture Notes in Com puter Science 2139, Springer, 2001, pp. 213–229.

[9] D. Boneh, and J. Shaw, *Collusion Secure Fingerprinting for Digital Data*, IEEE Transactions on Information Theory, Vol 44, No. 5, 1998, pp. 1897–1905.

[10] A. Brodnick and J. I. Munro, *Membership in Constant Time and Almost-Minimum Space,* SIAM J. Comput. 28(5), 1999, pp. 1627–1640.

[11] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, A. Sahai, *Exposure-Resilient Functions and All-or-Nothing Transforms,* Advances in Cryptology - EUROCRYPT 2000 Proceedings, Lecture Notes in Computer Science 1807, Springer, 2000, pp. 453–469.

[12] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, *Multicast Security: A Taxonomy and Some Efficient Constructions,* Proc. of Proceedings IEEE INFOCOM '99, Vol. 2, 1999, pp. 708–716,

[13] R. Canetti, T. Malkin, K. Nissim, *Efficient Communication-Storage Tradeoffs for Multicast Encryption,* Advances in Cryptology - EUROCRYPT'99 Proceedings, Lecture Notes in Computer Science 1592, Springer, 1999, pp. 459–474.

[14] B. Chor, A. Fiat and M. Naor, *Tracing traitors*, Advances in Cryptology - CRYPTO '94, Lecture Notes in Computer Science, Vol. 839, Springer, pp. 257–270, 1994.

[15] B. Chor, A. Fiat, M. Naor and B. Pinkas, *Tracing traitors*, IEEE Transactions on Information Theory, Vol. 46, No. 3, May 2000, pp. 893–910.

[16] C. Cocks, *An identity based encryption scheme based on quadratic residues*, Cryptography and Coding, Lecture Notes in Computer Science 2260, Springer, 2001, pp. 360–363.

[17] Content Protection for Recordable Media. Available:
http://www.4centity.com/4centity/tech/cprm

[18] 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, 1999, pp. 13–25.

[19] Z. J. Czech, G. Havas and B. S. Majewski, *Perfect Hashing*, Theoretical Computer Science 182, 1997, pp. 1–143.

[20] D. Dolev, C. Dwork and M. Naor, *Nonmalleable Cryptography,* SIAM J. Computing 30(2), 2000, pp. 391–437.

[21] C. Dwork, J. Lotspiech and M. Naor, Digital Signets: Self-Enforcing Protection of Digital Information, 28th Symposium on the Theory of Computation (1996), pp. 489– 498.

[22] P. Erdos and R. Rado, *Intersection Theorems for Systems of Sets,* Journal London Math. Soc. 35 (1960), pp. 85–90.

[23] U. Feige, P. Raghavan, D. Peleg, E. Upfal, *Computing with Noisy Information*, SIAM J. Comput. 23(5): 1001-1018 (1994)

[24] A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science 773, Springer, 1994, pp. 480—491.

[25] A. Fiat and T. Tassa, *Dynamic Traitor Tracing,* Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science, Vol. 1666, 1999, pp. 354–371.

[26] E. Fujisaki and T. Okamoto, *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, Advances in Cryptology - CRYPTO 1999, Lecture Notes in Computer Science, Vol. 1666, 1999, pp. 537–554.

[27] E. Gafni, J. Staddon and Y. L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*, Advances in Cryptology - CRYPTO 1999, Lecture Notes in Computer Science, vol. 1666, Springer, 1999, pp. 372–387.

[28] J.A. Garay, J. Staddon and A. Wool, Long-Lived Broadcast Encryption. Advances in Cryptology - CRYPTO'2000, Lecture Notes in Computer Science, vol 1880, pp. 333–352, 2000.

[29] O. Goldreich, S. Goldwasser and S. Micali, *How to Construct Random Functions,* J. of the ACM 33(4), 1986, pp. 792–807.

[30] S. Goldwasser and S. Micali. *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, April 1984, pp. 270–299.

[31] D. Halevy and A. Shamir, *The LSD Broadcast Encryption Scheme,* to appear, Advances in Cryptology - CRYPTO '2002, Lecture Notes in Computer Science, Springer, 2002.

[32] R. Kumar, R. Rajagopalan and A. Sahai, *Coding Constructions for blacklisting problems without Computational Assumptions,* Advances in Cryptology - CRYPTO '99, Lecture Notes in Computer Science, vol 1666, 1999, pp. 609–623.

[33] M. Luby and J. Staddon, *Combinatorial Bounds for Broadcast Encryption,* Advances in Cryptology - EUROCRYPT '98, Lecture Notes in Computer Science, vol 1403, 1998, pp. 512–526.

[34] 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). Available `www.csee.umbc.edu/~sherman/Rtopics/Crypto/oft.html/`.

[35] M. Naor, *String Matching with Preprocessing of Text and Pattern*, ICALP 1991 Proceeding, Lecture Notes in Computer Science 510, Springer, 1991, pp. 739–750.

[36] M. Naor, *Tradeoffs in Subset-Cover Revocation Schemes*, manuscript, 2001.

[37] M. Naor and B. Pinkas, *Threshold traitor tracing*, Advances in Cryptology - Crypto '98, Lecture Notes in Computer Science, Vol. 1462, pp. 502–517.

[38] M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes,* Financial Cryptography, 4th International Conference, FC 2000 Proceedings, Lecture Notes in Computer Science 1962, Springer, 2001, pp. 1–20.

[39] M. Naor, B. Pinkas and O. Reingold, *Distributed Pseudo-random Functions and KDCs,* Advances in Cryptology -EUROCRYPT 1999, Lecture Notes in Computer Science, vol. 1592 Springer, 1999, pp. 327–346.

[40] M. Naor and O. Reingold, *Number-theoretic Constructions of Efficient Pseudo-random Functions*, Proc. of 38th IEEE Symposium on Foundations of Computer Science, 1997, pp. 458–467.

[41] R. Pagh, Low redundancy in static dictionaries with O(1) lookup time, Proceedings of ICALP '99, Lecture Notes in Computer Science 1644, Springer, 1999, pp. 595–604.

[42] B. Pfitzmann, Trials of Traced Traitors, Information Hiding Workshop, First International Workshop, Cambridge, UK. Lecture Notes in Computer Science, Vol. 1174, Springer, 1996, pp. 49–64.

[43] R. L. Rivest, All-or-Nothing Encryption and the Package Transform. Proc. 4th Fast Software Encryption International Workshop, 1997, Lecture Notes in Computer Science, Vol. 1267, Springer, 1997, pp. 210–218.

[44] R. Safavi-Naini and Y. Wang, Sequential Traitor Tracing Advances in Cryptology - CRYPTO 2000, Lecture Notes in Computer Science, vol 1880, 2000, pp. 316–332.

[45] V. Shoup and R. Gennaro, *Securing threshold cryptosystems against chosen ciphertext attack*, Advances in Cryptology - EUROCRYPT '98, Lecture Notes in Computer Science 1403, Springer, 1998, pp. 1–16.

[46] D.R. Stinson and R. Wei, *Key Preassigned Traceability Schemes for Broadcast Encryption*, Proc. Fifth Annual Workshop on Selected Areas in Cryptography, Lecture Notes in Computer Science, Vol. 1556 (1999), pp. 144–156.

[47] M. Waldvogel, G. Caronni, D. Sun, N. Weiler and B. Plattner, *The VersaKey Framework: Versatile Group Key Management,* IEEE Journal on Selected Areas in Communications, vol 17 no. 9, Septemeber 1999.

[48] 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

[49] C. K. Wong, M. Gouda and S. Lam, Secure Group Communications Using Key Graphs, SIGCOMM 1998.