The LSD Broadcast Encryption Scheme

Dani Halevy and Adi Shamir

Applied Math. Dept.
The Weizmann Institute of Science
Rehovot 76100, Israel
{danih,shamir}@wisdom.weizmann.ac.il

Abstract. Broadcast Encryption schemes enable a center to broadcast encrypted programs so that only designated subsets of users can decrypt each program. The stateless variant of this problem provides each user with a fixed set of keys which is never updated. The best scheme published so far for this problem is the "subset difference" (SD) technique of Naor Naor and Lotspiech, in which each one of the n users is initially given $O(\log^2(n))$ symmetric encryption keys. This allows the broadcaster to define at a later stage any subset of up to r users as "revoked", and to make the program accessible only to their complement by sending $O(r)$ short messages before the encrypted program, and asking each user to perform an $O(log(n))$ computation. In this paper we describe the "Layered Subset Difference" (LSD) technique, which achieves the same goal with $O(\log^{1+\epsilon}(n))$ keys, $O(r)$ messages, and $O(\log(n))$ computation. This reduces the number of keys given to each user by almost a square root factor without affecting the other parameters. In addition, we show how to use the same LSD keys in order to address any subset defined by a nested combination of inclusion and exclusion conditions with a number of messages which is proportional to the complexity of the description rather than to the size of the subset. The LSD scheme is truly practical, and makes it possible to broadcast an unlimited number of programs to 256,000,000 possible customers by giving each new customer a smart card with one kilobyte of tamper-resistant memory. It is then possible to address any subset defined by t nested inclusion and exclusion conditions by sending less than $4t$ short messages, and the scheme remains secure even if all the other users form an adversarial coalition.

1 Introduction

Broadcast Encryption schemes enable a center to deliver encrypted data to a large set of users so that only a particular subset of *privileged users* can decrypt it. Such schemes are useful in pay-TV systems, the distribution of copyrighted material on encrypted CD/DVD disks, internet multicasting of video music and magazines, the distribution of commercial catalogs and price lists on a need-to-know basis, etc.

This basic problem has many variants. For example, the privileged sets can be arbitrary, size-limited, or with tree-like structure. They can be fixed, slowly

M. Yung (Ed.): CRYPTO 2002, LNCS 2442, pp. 47–60, 2002.

changing, or rapidly changing. The scheme can be used to support a single, bounded, or unbounded number of broadcasts. The keys stored by each user can be fixed, time-dependent, or modifiable by previous transmissions. User revocation can be permanent or limited to a single program. The scheme can be resistant to random or arbitrary adversarial coalitions of various sizes. And so on.

The most interesting (and arguably the hardest) variant of broadcast encryption deals with *stateless receivers* and has the following requirements:

- Each user is initially given a collection of symmetric encryption keys.
- The keys can be used to access any number of broadcasts.
- The keys can be used to define any subset of users as privileged.
- The keys are not affected by the user's viewing history.
- The keys do not change when other users join or leave the system.
- Consecutive broadcasts can address unrelated privileged subsets.
- Each privileged user can decrypt the broadcast by himself.
- Even a coalition of all the non-privileged users cannot decrypt the broadcast.

Early papers on the topic concentrated on general key management issues in star shaped networks. In 1991, S. Berkovits published an article, "How to Broadcast a secret"[1], in which he presented several broadcast schemes based on secret sharing (see [9]). However, his matrix based schemes were impractical for large sets of users and insecure under repeated use.

In 1994, Moni Naor and Amos Fiat[5] formalized the basic definitions and paradigms of this field. In particular, they presented schemes in which each user has a fixed reusable set of keys. However, the complexity of their schemes was strongly dependent on the size k of the adversarial coalition, and their best result required storage of o(klog(k) log(n)) keys per user and transmision of $o(k^2(\log^2(k))\log(n))$ messages by the broadcaster. These complexities are too high in applications such as pay-TV in which thousands of smart cards can be obtained and analysed by commercial pirates.

A simple solution to the broadcast encryption problem is to give each user u a unique symmetric key K_u which is known only to the broadcaster and the user. The program is broadcast multiple times, encrypted under the key of each privileged user. To save bandwidth, we can use the broadcast encryption scheme only in order to predeliver a short program key K to the privileged subset. The actual program is broadcast only once, encrypted by this K. This scheme requires $O(1)$ storage and processing per user, but the transmission length is $O(s)$ where s is the size of the privileged set. Consequently, this scheme is practical only when the privileged sets are small (or slowly expanding if the broadcaster reuses old program keys and the privileged users memorize them).

A folklore extension of this technique is to consider the n users as the leaves of a balanced binary tree of height $\log(n)$. A unique key is assigned to each vertex in this tree, and each user knows the $\log(n)$ keys of all its tree ancestors. This makes it possible to use a single message in order to address a complete subtree of privileged users, and the total number of messages in the general case is proportional to the number of subtrees required to cover the privileged subset. To minimize the average number of subtrees, the broadcaster should assign users to leaves based on their similarity rather than in random order.

This tree representation makes it possible to revoke any single user u by describing all the other users as a union of the $\log(n)$ subtrees that "hang off" the tree path from the root to u. To address them, the broadcaster sends $\log(n)$ short messages which contain the program key encrypted under the unique key of each one of the roots of these subtrees. An extension of this technique (which is called *the complete subtree method* in [6]) makes it possible to simultaneously revoke any r users with $r\log(n/r)$ messages, and the scheme is secure against any coalition of revoked users.

A major improvement of this idea was the "subset difference" (SD) technique developed in Naor Naor and Lotspiech (NNL[6]), which is the current state of the art in stateless broadcast schemes. In the SD algorithm, the number of keys given to each user is $O(\log^2(n))$ (which depends only on the total number of users), and the number of messages is $O(r)$ (which depends only on the number of revoked users). Each privileged user has to perform $O(log(n))$ cryptographic operations in order to retrieve the program key, and the scheme can be used an arbitrary number of times with arbitrary subsets of revoked users.

In this paper we improve and generalize this technique. We introduce the "Layered Subset Difference" (LSD) scheme, and show that for any $\epsilon > 0$ we can create a stateless broadcast encryption scheme with $O(\log^{1+\epsilon}(n))$ keys, $O(r)$ messages, and $O(log(n))$ cryptographic operations. The improved scheme is truly practical: By using less than one kilobyte of storage in each user's smart card, the broadcaster can revoke any r out of $2^{28} \approx 256,000,000$ possible users by sending at most $4r$ messages. The actual number of messages is typically smaller than this worst case bound, and initial experiments indicate that for random subsets of r revoked users, the average number of messages is approximately $2r$.

In the last part of this paper, we generalize the scheme by considering more complicated types of privileged sets defined by nested inclusion and exclusion conditions. Such a representation is common in legal documents (which initially define the general rule, then list the exceptions, then list exceptions to the exceptions, and so on). As a typical example, consider a satellite TV operator that wants to broadcast a baseball event which is held in Boston. In general, he wants to make the program accessible to all its customers in New England who subscribe to the sport channel, except for those who live in Boston (in order to encourage local ticket sales). However, within Boston the game should be available to sport bars (which have special subscriptions) except for Joe's Bar and Moe's Bar (who stopped paying their fees last month). If the customers are organized in a tree-like structure based on their geographic location and type of subscription, the broadcaster can describe this complex set with a small number of nested inclusion and exclusion conditions on the vertices of the tree. We show that we can use the same LSD keys in order to address any set defined by t conditions with an essentially optimal number of $O(t)$ messages. Since the number of messages in our scheme depends only on the difficulty of describing the privileged set and not on its size or the total number of customers, it can be orders of magnitude more efficient than previous algorithms which try to individually list or revoke millions of users.

2 Improvement: The Layered Subset Difference Scheme

The basic idea in all the stateless broadcast encryption schemes is to represent any privileged set as the union of s subsets of users of a particular form. A different key is associated with each one of these sets, and a user knows a key if and only if he belongs to the corresponding set. The broadcaster encrypts the program key s times under all the keys associated with the sets in the cover. Consequently, each privileged user can easily access the program, but even a coalition of all the non-privileged users cannot find the program key.

The simplest implementation of this idea is to cover the privileged set with singleton sets. A better solution is to associate the users with the leaves of a binary tree, and to cover the privileged set of leaves with a collection of subtrees. However, these covering strategies are inefficient when the privileged set is the complement of a small number of revoked users.

The improved performance of the SD algorithm is primarily due to its more sophisticated choice of covering sets:

Definition 1. *Let i be any vertex in the tree and let j be any descendant of i. Then $S_{i,j}$ is the subset of leaves which are descendants of i but are not descendants of j.*

Note that $S_{i,j}$ is empty if $i = j$. Otherwise, $S_{i,j}$ looks like a tree with a smaller subtree cut out. An alternative view of this set is as a collection of subtrees which are hanging off the tree path from i to j.

The SD scheme covers any privileged set P defined as the complement of r revoked users by the union of $O(r)$ of these $S_{i,j}$ sets. What we show in this paper is that this collection of sets can be drastically pruned: A small subcollection of the $S_{i,j}$ sets suffices to represent any such P as the union of $O(r)$ of the remaining sets, with a slightly larger constant. Since there are fewer possible sets, we can reduce the number of initial keys given to each user. We first show that if we allow the number of sets in the cover to grow by a factor of two, we can reduce the number of keys from $O(\log^2(n))$ to $O(\log^{3/2}(n))$, and then we extend the technique and show how to reduce the number of keys to $O(\log^{1+\epsilon}(n))$ for any fixed $\epsilon > 0$.

2.1 The Original SD Scheme

In this section we describe only those parts of the Subset Difference scheme which are required in order to understand our improvement. Further details and discussions can be found in the NNL paper.

During the initialization stage, the broadcaster defines a binary tree with n leaves associated with the users. To simplify our notation, we assume that the tree is balanced and that $log_2(n)$ is an integer. The broadcaster associates with each non-leaf i an m bit label which is chosen randomly and independently. A pseudo random generator G extends an m bit input x into a $3m$ bit output, which is used to define new labels and keys recursively for all the nodes j in the subtree whose root is i in the following way: Given the label x of vertex j, use the three parts of $G(x)$ to define the key $K_{i,j}$ associated with the set $S_{i,j}$, the label of the left child of j, and the label of the right child of j. Note that each vertex j gets multiple labels derived from the different starting points i above it, and each one of them yields a different key. The crucial property of this derivation technique is that given the label of vertex j derived from starting point i, it is easy to compute the key of any set $S_{i,j'}$ in which j' is a descendant of j, but it is infeasible to compute any other keys (whose sets either have a different i or the same i but a j' which is not a descendant of j).

The number of nonempty sets $S_{i,j}$ that contain a particular leaf u is $O(n)$. User u cannot store so many independent keys in a large system with millions of users, but due to their interdependence he can derive all of them from the smaller set of labels in which i is an ancestor of u and j is just off the tree path from i to u (at distance 1). The number of these labels is $O(\log^2(n))$, and u is entitled to know all of them since he belongs to all these sets $S_{i,j}$.

To revoke r users, the broadcaster uses the following covering strategy: As long as there are at least two revoked users in the tree, choose a smallest subtree with this property, and denote its root by v. Add to the cover the sets $S_{k,l}$ and $S_{p,q}$ where k is the left child of v, l is the revoked leaf in the subtree rooted at k, p is the right child of v and q is the revoked leaf in the subtree rooted at p. (Note that if $k = l$ or $p = q$ the corresponding set is empty, and there is no need to add it to the cover). Delete from the tree all the vertices beneath v, mark it as revoked, and repeat the process. If we end with the root marked as revoked we are done, otherwise we add to the cover the set $S_{r,v}$ where r is the root and v is the single remaining revoked vertex, and stop.

The number of sets in the resultant cover is at most 2r-1 since at each step we pick at most 2 subsets and decrease the number of revoked leaves by 1. So, before the last step we pick at most 2r-2 subsets, and in the last step we pick at most one additional subset. Most of the splitting is expected to happen at the top of the tree, and thus many subsets from that region are likely to be empty. For a random choice of r revoked leaves, the average number of sets in the cover was experimentally found to be approximately $1.25r$.

Figure 1 demonstrates the seven subsets actually picked by the SD algorithm in a tree with 4 revoked users v_6, v_9, v_{13}, v_{14}. It suffices to consider their Steiner tree, which is the minimal subtree which contains all of them and the root. A thick edge represents the "descendant of" relation (e.g., $v2$ is a descendant but not necessarily a direct child of $v1$), whereas a thin edge represents the "child of" relation (e.g., $v3$ is a child of $v2$). The sets picked by the algorithm described above are actually all the subsets $S_{i,j}$ where $parent(i)$ is either undefined (when i is the root) or a splitting point in the Steiner tree, and j is the nearest splitting point underneath i. These sets are denoted by the double pointed arcs, and each

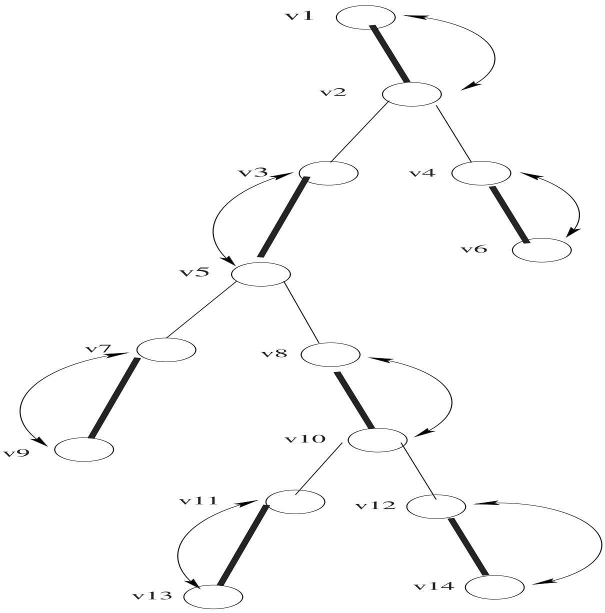

Fig. 1. A Steiner tree

one of them contains all the leaves of the original tree which hang off the chain of edges in the Steiner tree represented by the thick line.

2.2 The Basic LSD Scheme

In this section we describe the simplest version of the Layered Subset Difference scheme. Our main observation is the following Lemma:

Lemma 1. *If i, k, j are vertices which occur in this order on some root-to-leaf path in the tree, then $S_{i,j}$ can be described as the disjoint union $S_{i,j} = S_{i,k} \cup S_{k,j}$*

Proof: $S_{i,k}$ is defined as a set difference $A \setminus B$ and $S_{k,j}$ is defined as a set difference $B \setminus C$ for some sets of leaves that satisfy $A \supseteq B \supseteq C$. The sets $S_{i,k}$ and $S_{k,j}$ are clearly disjoint, and their union is $A \setminus C$ which is the definition of $S_{i,j}$. Note that if k is not on the path from i to j, and we use the natural extension of the definition of these sets, the lemma is incorrect. For example, if k is the root and i, j are distinct leaves then the set of leaves below i but not below j is $\{i\}$, the set of leaves below i but not below k is empty, the set of leaves below k but not below j contains all the leaves except j, and these sets do not satisfy the desired relationship. $\diamond$

The basic idea of the LSD scheme is to retain only a small subcollection of the $S_{i,j}$ sets used by the SD scheme. Whenever the broadcaster wants to use a discarded set in his set cover, he replaces it by the union of two smaller sets which are in the subcollection.

The subcollection of sets $S_{i,j}$ in the LSD scheme is defined by restricting the levels in which the vertices i and j can occur in the tree. We define some of the $\log(n)$ levels as "special". The root is considered to be at a special level, and in addition we consider every level of depth $k \cdot \sqrt{\log(n)}$ for $k = 1..\sqrt{\log(n)}$ as special (wlog, we assume that these numbers are integers). There are thus $\sqrt{\log(n)}$ special levels which are equally spaced at a distance of $\sqrt{\log(n)}$ from each other. The collection of levels between (and including) adjacent special levels is defined as a "layer".

The subcollection of sets in the LSD scheme is defined in the following way:

Definition 2. *$S_{i,j}$ is a useful set if it is not empty, and at least one of the following conditions is true: both i and j belong to the same layer, or i is at a special layer.*

Lemma 2. *Any nonempty set $S_{i,j}$ is either a useful set or the disjoint union of two useful sets.*

Proof: Since $S_{i,j}$ is nonempty, j is a strict descendant of i. If they belong to the same layer, then $S_{i,j}$ is a useful set. Otherwise, define k as the first vertex on the path from i to j which is in a special level (possibly i itself). Since i and k are in the same layer, $S_{i,k}$ is a useful set (unless it is empty). Since k is in a special level, $S_{k,j}$ is useful (unless it is empty) even if k and j belong to different layers. Consequently, $S_{i,j}$ is the disjoint union of the two useful sets $S_{i,k}$ and $S_{k,j}$, or equal to one of them if the other is empty. $\diamond$

Since the number of covering sets in the SD scheme is bounded by $2r - 1$ and each one of them is replaced by at most two useful sets during the actual broadcast, the number of messages sent by the broadcaster in this scheme is at most $4r - 2$. The $1.25r$ average complexity for r randomly chosen revoked users in the SD scheme suggests an $2.5r$ average complexity in the modified scheme, but in fact there is an additional saving since many of the sets do not get split. Actual experiments on trees with 200,000 users indicate that the average complexity is closer to $2r$, which is only 1.6 times larger than in the original SD scheme.

What we gain from this slightly increased message complexity is a significant reduction in the size of the tamper resistant memory in each user's smart card:

Lemma 3. *The number of labels memorized by each user u in the basic LSD scheme is $O(\log^{3/2}(n))$.*

Proof: The keys associated with the broadcast sets are generated by the user in the same recursive way as in the SD algorithm. The only labels a user should memorize are those that correspond to sets $S_{i,j}$ in which i is an ancestor of u, j is just hanging off the path from i to u, and the levels of i and j are those specified in the definition of useful sets. Note that at each level in the tree there can be at most one vertex which can serve as i and one vertex which can serve as j wrt u, and these two vertices are siblings.

To count the number of memorized labels, consider the two possible cases of useful sets:

- Local sets: Each layer contains $\sqrt{\log(n)}$ levels, and thus the number of i and j pairs which belong to that layer is $O((\sqrt{\log(n)})^2) = O(\log(n))$. Since there are $\sqrt{\log(n)}$ possible layers, the number of i and j pairs of this type is $O(\log^{3/2}(n))$.
- Special sets: Each i in a special level can be associated with a j in any one of the $O(\log(n))$ levels underneath it. Since there are $\sqrt{\log(n)}$ possible i's, the number of labels of this type is also $O(\log^{3/2}(n))$.

Consequently, the total number of labels u has to know is reduced from $O(\log^2(n))$ to $O(\log^{3/2}(n))$. It is easy to show that the choice of $\sqrt{\log(n)}$ as the distance between consecutive special levels is optimal among all the equidistant partitions. For any choice of distance s between consecutive special levels, the number of keys each user has to remember is $O(s^2 \times (\log(n)/s) + (\log(n) \times \log(n)/s))$. The first derivative of this expression(as a function of s) is $\log(n) - \log^2(n)/s^2$ which is equal to 0 for $s = \sqrt{\log(n)}$. Since the second derivative is positive, choosing this value of s minimizes the storage complexity of this scheme. $\diamond$

2.3 The General LSD Scheme

In this section we show how to further reduce the memory requirements of the user revocation scheme, by solving an interesting graph theoretic problem.

The basic LSD algorithm represents each $S_{i,j}$ as the disjoint union of two sets from a smaller subcollection. It is easy to generalize this observation and represent $S_{i,j}$ as the disjoint union of d sets:

Lemma 4. *Let $i, k_1, k_2, \ldots, k_{d-1}, j$ be any sequence of vertices which occur in this order (but not necessarily consecutively) along some root-to-leaf path in the tree. Then*

$$S_{i,j} = S_{i,k_1} \cup S_{k_1,k_2} \cup \cdots \cup S_{k_{d-1},j}$$

Proof: This is just a telescoping formula of set differences for any descending chain of sets. $\diamond$

Any root to leaf path can be viewed as a line graph of length $\log(n)$ with directed edges between adjacent vertices. Broadcasting the set $S_{i,j}$ corresponds to walking from vertex i to vertex j, and addressing all the subtrees that hang off this segment. The original line graph has very few edges (whose labels require very little memory) but these edges provide only a slow way of walking from i to j (with many messages). Since for each original edge the corresponding set is a single subtree, this covering technique is equivalent to the Complete Subtree method of [6].

The Subset Difference technique adds to the line graph all the edges in its directed transitive closure. We can now jump from any i to any descendant j in a single jump (and thus address all the users in $S_{i,j}$ with a single message), but each user has to memorize the labels of $O(\log^2(n))$ edges.

The basic LSD scheme shows that we only have to use $O(\log^{3/2}(n))$ edges (labels) in order to get from any i to any descendant j in two steps (messages). The general LSD scheme considers the following graph theoretic problem: What is the smallest number of edges we have to add to the line graph in order to guarantee the existence of a directed path of length at most d from any i to any descendant j?

To make the graph construction applicable to our user revocation problem, we have to add two additional constraints:

- Monotonicity: We can only add an edge from i to one of its descendants.
- Shrinkage: If we add an edge from i to j, we also have to add all the edges from i to vertices j' between i and j.

Note that without the monotonicity condition, we can get in two steps from any i to any j by adding the $2\log(n)$ directed edges from each vertex to the root and from the root to each vertex. However, such a nonmonotonic path does not correspond to a legal set partition. The shrinkage condition is required in order to guarantee that the shrunk versions of sets provided to users (in which we stop at the first j' on the path from i to j that hangs off the path from the root to the user) also belong to the subcollection. To see the importance of this condition, consider the basic LSD construction. Our original definition of useful sets was closed under shrinkage. However, we could have used the following alternative definition: $S_{i,j}$ is useful if i and j belong to the same layer, or j is in a special layer. We can still get from any i to any descendant j in two steps (through the last special vertex k on the path from i to j), and the number of sets of this type is still $O(\log^{3/2}(n))$. However, this collection of sets is not closed under shrinkage, and thus we have to give each user the labels of all the $O(\log^2(n))$ possible $S_{i,j'}$ sets (where neither i nor j' are special), since each one of them can be the shrunk version of some useful set $S_{i,j}$ that will be used by the broadcaster.

We describe an efficient solution to this graph theoretic problem by representing each vertex on the line graph by its distance from the root, expressed as a d digit number in base $b = O(log^{1/d}(n))$. The root is represented by $0\ldots00$, its child is represented by $0\ldots01$, etc.

Our goal is to define a small subcollection of *useful transformations* between pairs of numbers which satisfy the monotonicity and shrinkage condition, and allow us to change any i to any larger j with a sequence of at most d useful transformations. Consider for example the problem of changing $i = 825917$ to $j = 864563$ in standard decimal notation. The simplest solution is to allow arbitrary single-digit transformations such as

$$825917 \longrightarrow 865917 \longrightarrow 864917 \longrightarrow 864517 \longrightarrow 864567 \longrightarrow 864563$$

However, these transformations do not satisfy either the monotonicity or the shrinkage condition. Consequently, we have to use a more complicated sequence of transformations:

Definition 3. *Let i be represented as a d digit number in base b by $\vec{x}a\vec{0}$ where a is the rightmost nonzero digit, $\vec{x}$ is a sequence of arbitrary digits, and $\vec{0}$*

is a sequence of zeroes. The transformation of i to j is called useful if j is represented either by $\overrightarrow{x+1}0\overrightarrow{0}$ or by any number $\overrightarrow{x}a'\overrightarrow{y}$ in which $a' \geq a$ and $\overrightarrow{y}$ is an arbitrary sequence of digits of the same length as $\overrightarrow{0}$.

The basic LSD scheme can be viewed as a special case of this definition for $d = 2$, where two digit numbers ending with 0 are considered to be special. In the general LSD scheme the number of trailing zeroes in the representation of i determines how special it is and how big is the layer within which it is allowed to jump (j can be any destination between $i + 1$ and the first vertex which is even more special than i, inclusive). In our previous example, these useful transformations allow the broadcaster to split the $S_{i,j}$ set into the following segments:

$$825917 \longrightarrow 825920 \longrightarrow 826000 \longrightarrow 830000 \longrightarrow 864563$$

Note that from the point of view of the broadcaster, the first three transitions are of the first type (which jumps to the end of the layer and increases the specialty level of the vertex), and the last transition is of the second type (which jumps to the middle of the layer). However, from the point of view of the user he knows the label associated with at most one of these transitions, and its memorized shrunk version is likely to be of the second type even if the broadcast transition was of the first type.

To count the number of useful transformations, consider any pair of i and j linked by a single useful transformation. We can choose the location of the digit a within i in d ways, and for each location we can choose the $d - 1$ digits in the sequences $\overrightarrow{x}$ and $\overrightarrow{y}$ in b^{d-1} ways, and the two digits $a \leq a'$ in $b^2/2$ ways. Since $b = O(\log^{1/d}(n))$, we get a total number of useful transformations of $O(d \cdot b^{d+1}) = O(d \cdot \log^{(d+1)/d}(n))$. This is an upper bound on the number of labels each user has to memorize, and the corresponding bound on the number of broadcast messages is $d(2r-1)$. Note that our construction of useful transformations is clearly optimal for regular graphs: To reach every vertex within d steps in a regular graph of size e and degree f, the condition $f^d \geq e$ must be clearly satisfied. For our parameters the degree f must satisfy $f \geq O(\log^{1/d}(n)))$, and our construction has such an indegree for any j. Noga Alon(private communication) has recently proved that this lower bound applies to all graphs(regular and nonregular) and thus our broadcast encryption scheme cannot be further improved by analyzing this graph theoretic problem. To find the asymptotic complexity of this general LSD scheme, choose d as a large enough constant. The message complexity remains $O(r)$, while the number of memorized labels is reduced to $O(\log^{1+\epsilon}(n))$ for any $\epsilon > 0$. Alternatively, if we use binary numbers with $d = \log\log(n)$ bits to represent each one of the $log(n)$ levels in the tree, we get another interesting tradeoff point of $O(r \cdot \log\log(n))$ messages and $O(\log(n) \cdot \log\log(n))$ storage per user.

The reader should be warned that these extended LSD schemes are mostly of theoretical interest. The basic LSD algorithm is significantly better than the original SD algorithm for practical values of n, but the additional improvements are noticable only for an astronomical number of users.

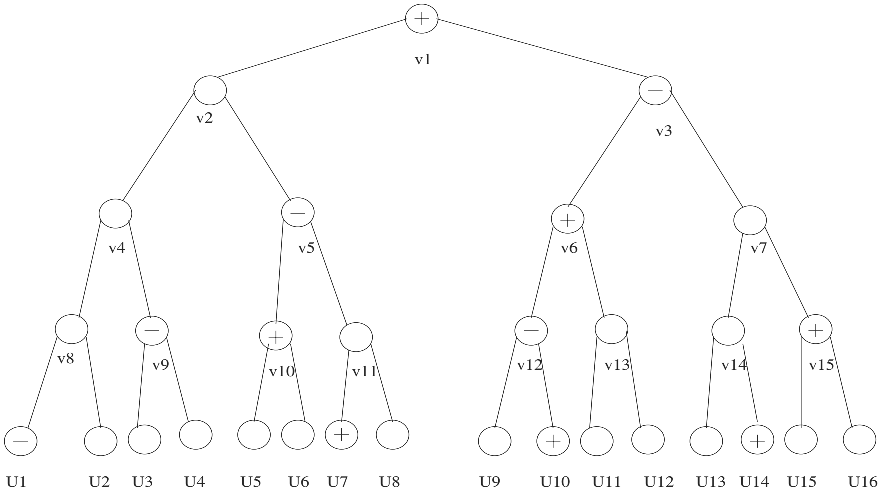

Fig. 2. A typical IE-tree

3 Generalization: Addressing IE-Trees

Standard broadcast encryption schemes deal with the problem of either addressing or avoiding a given list of users, and their complexity depends on the size of this list. In this section we consider a new way of defining the privileged sets, which allows the broadcaster to define large complex sets in a natural way by nesting multiple inclusion and exclusion conditions on the vertices of the user tree. The corresponding data structure is called an IE-tree (where IE stands for Inclusion Exclusion), and a typical example of such a tree appears in Fig. 2. Each vertex in the tree is either unmarked, marked with +, or marked with -. We assume that the root is marked, and thus for any leaf u there is a well defined closest (i.e., lowest, or most specific) marked vertex on its path to the root. The mark of this vertex determines whether u is in the privileged set or not. For example, in Fig. 2 the privileged set consists of the leaves $u_2, u_5, u_6, u_7, u_{10}, u_{11}, u_{12}, u_{14}, u_{15}$ and u_{16}, whereas the non-privileged set consists of u_1, u_3, u_4, u_8, u_9 and u_{13}.

An IE-tree is a very compact way of representing interesting subsets of users, provided we assign users to leaves in a meaningful order (e.g., by geographic location, type of subscription, type of smart card, profession, etc). Fortunately, the SD and LSD schemes can efficiently deal with such a generalized definition of privileged sets, with only minor changes.

Given an IE-tree, we can assume WLOG that the root is marked, and all the marked vertices along any root to leaf path in the tree contain alternating marks (if two vertices along such a path are marked in the same way and there is no opposite mark between them, the lower mark is redundant and can be eliminated). We then apply the following iterative procedure:

If there are any vertices marked by +, find a lowest such vertex i. The subtree below i is either completely unmarked, or it contains some - signs. In the first case, if i is the root then stop, otherwise add the subset $S_{parent(i),sibling(i)}$ to the set cover. In the second case, apply the user revocation scheme to the subtree rooted at i. In either case, eliminate all the signs in the subtree (including the + sign of vertex i), and repeat the process.

To show the correctness of this algorithm, we use the fact that if the subtree rooted at i has no - signs we want to address all its leaves, and if it has - signs then the SD or LSD schemes choose messages that address exactly the privileged users in this subtree. To show that the sign elimination rule is correct, we note that i is marked by +. Since signs are alternating, there must be a - sign above i (unless it is the root, in which case we stop). When we eliminate all the signs from the subtree, all the leaves in this subtree automatically change their status to non-privileged due to the - sign above i, and thus the rest of the algorithm (assuming that it works correctly) will consider the whole subtree as revoked and will not try to address any one of its leaves. If we wish, we can eliminate the whole subtree rooted at i (and not just its signs) from the original tree, since it will not be considered by the rest of the algorithm. The broadcast messages correspond to all the sets $S_{i,j}$ chosen during any one of the phases of the algorithm.

To count the number of messages broadcast by an algorithm which uses the SD revocation scheme, we note that for every vertex i marked by + we either pick one subset (if i is the only marked vertex in its tree) or at most $2r_i$ subsets where r_i is the number of vertices marked by '-' in this tree, and then we eliminate all these marks. Hence, the number of picked subsets is bounded by $2M + P$ where M is the number of vertices marked by - and P is the number of vertices marked by + in the original IE-tree. When we use the basic LSD revocation scheme, we get a slightly larger message bound of $4M + P$, but each user has to memorize a considerably smaller number of labels.

Since there can be several meaningful ways in which users should be assigned to the leaves of the tree, and the broadcaster may want to use different orders on different occasions, he can use as his data structure the extended notion of an *IE-multitree*, defined as a collection of h independent trees which share their leaves (in possibly different orders). A simple example of a multitree consisting of two trees with four common leaves is described in Fig 3. Each user gets h sets of labels (one per tree), and thus the broadcaster can address users purely by geographic location in one program, and purely by their type of subscription in a second program. While it is possible to use different levels in a single tree to partition users by different types of criteria, the number of marked vertices (and thus the message complexity) for typical privileged sets can be much higher than in the IE-multitree representation.

4 Final Comments

4.1 Security Analysis

Since in all the variations of the LSD scheme each user stores only a subset of the labels he stores in the original SD algorithm, the power of each coalition (in terms of the keys they know) can only diminish, and thus we can use exactly

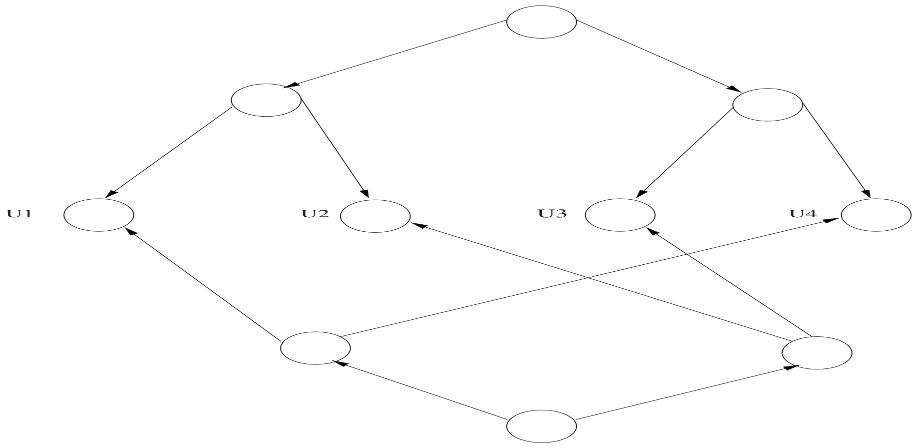

Fig. 3. An IE-multitree

the same argument as in the original NNL paper to show that broadcasts remain inaccessible even to a coalition of all the non-privileged users, under the assumption that the generator G and the symmetric encryption scheme are cryptographically secure. Note that if we don't need this strong notion of security and only want to protect the broadcast from a single non-privileged user, there are simpler and more efficient constructions.

4.2 A Practical Example

Consider a large broadcasting operation with $2^{28} \approx 256,000,000$ users (which is comparable to the number of TV or Internet users in the US). These users can be represented by a complete binary tree with 28 levels, and we split them in the basic LSD scheme into five layers of sizes $6, 6, 6, 5$ and 5. Each user has to memorize 81 labels associated with special i's and 65 labels associated with non-special i's. The total number of labels is thus 146, and if each label is a 7-byte DES key they all fit into less than one kilobyte of memory. In the original SD scheme, each user has to memorize $28 * 29/2 = 406$ keys which require almost three kilobytes of memory. This difference is particularly significant if the broadcaster wants to use IE-multitrees with several sets of memorized labels which correspond to several selection criteria (one tree based on geographic location, another based on type of subscription, etc.). With the same memory size, smart cards using the LSD scheme can use three times more useful criteria than smart cards using the SD scheme. Note that the actual algorithm executed by the user is exactly the same in the SD and LSD schemes: check whether for one of the broadcast (i, j) pairs you are a descendant of i but not of j (there can be at most one such pair), and if so, derive the corresponding key by using the generator G at most 27 times (which requires negligible time). The only difference between the SD and LSD schemes is in the broadcaster's algorithms for choosing the user's keys and the subset covers.

A useful property of these algorithms is that it is not necessary to place all the users at the same level in the binary tree. A broadcaster can start with a small number of users placed high in the tree, and place new users further down when their numbers justify it. There is no need to replace the smart cards of existing users when the broadcaster opens up additional levels for new users, since the broadcaster can simultaneously address users at different levels in the tree. Similarly, there is no need to replace the smart cards of existing users when other users leave the system since the broadcaster can simply revoke them in his messages.

References

1. S. Berkovits, How to Broadcast a secret, Advances in Cryptology - Eurocrypt'91,Lecture Notes in Computer Science 547, Springer, 1991, pp.536-541.
2. Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, Benny Pinkas, Multicast Security: A Taxonomy and Some Efficient Constructions.
3. E. Gafni, J.Staddon and Y.L. Yin, Efficient methods for integrating traceability and broadcast encryption,Proc. Advances in Cryptology - Crypto '99, LNCS 1666, Springer, 1999, 372-387.
4. 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.
5. M. Naor, A. Fiat, Broadcast Encryption, Advances in Cryptology - Crypto 93',Lecture Notes in Computer Science 773, Springer, 1994, pp. 480-491.
6. D. Naor., M. Naor, J. Lotspiech, Revocation and Tracing Schemes for Stateless Receivers. February, 2001.
7. M. Naor, B. Pinkas, Threshold Traitor Tracing, Crypto 98.
8. M. Naor, B. Pinkas, Efficient Trace and Revoke Schemes, FC'2000.
9. Shamir, A., "How to Share a Secret", Communications of the ACM, vol. 22, NO. 11, November 1979, pp. 612-613.