A Survey of Broadcast Encryption

Jeremy Horwitz

13 January 2003

Abstract

Broadcast encryption is the problem of a sending an encrypted message to a large user base such that the message can only be decrypted by a dynamically changing privileged subset. The study of broadcast encryption has become more and more important with the ever-increasing concern about copyright issues and the increasing interest in secure multicasting (over cable television and the Internet). We discuss the early broadcast-encryption results of Fiat and Naor [3], presenting several broadcast-encryption schemes from that paper. Next, we discuss Naor, Naor, and Lotspiech's [6] subset-cover model, an abstract framework for broadcast-encryption schemes. We then discuss instances of the subset-cover model. This is followed by discussion of a method of Halevy and Shamir [4] that improves upon [6]'s best example of a subset-cover scheme. We finish with a discussion of traitor tracing — the problem of, given an illegal decoder box, punishing the users who contributed keys to it. In particular, we focus on traitor-tracing algorithms for (some) subset-cover schemes.

1 Introduction

A broadcast-encryption scheme is a collection of algorithms that allow a centralized transmitter to send encrypted messages to a collection of users such that only a (possibly changing) privileged subset of users can decrypt them. Such schemes have myriad applications, most notably the secure distribution of copyrighted media (movies on DVD, music on CD, books on CD-ROM, etc.) and the multicasting of such media over the Internet or over cable television.

We focus primarily on three papers. The first [3] laid the foundations for broadcast encryption, introducing the first formal study of the topic. These results are in terms of a factor k (as well as the number of total users), the number of colluding revoked users that the system is secure against. In these results, k must be chosen before the initial setup of the algorithms. The second paper [6] is not limited by this factor k; it presents its results in terms of the (possibly changing) total number of revoked users (as well as, again, the number of total users). An extremely general framework (the subset-cover model) for broadcast-encryption schemes is presented, as well as two specific schemes that fall under this framework. The third paper [4] presents several improvements upon one of the subset-cover schemes of [6].

Also of interest is the related process of traitor tracing: given an illegal decoder box, determine which privileged user(s) supplied the keys necessary for the box's construction (technically, the algorithm presented in [6] only guarantees to make the traitors' boxes useless). We discuss results in the black-box model, where it is assumed that the box cannot be opened, *i.e.*, the box can only be used by giving it an input (supposedly an encrypted message) and examining its output (the decryption of that message).

We will continue in Section 2 with various preliminaries including important definitions that will be used throughout the paper. Section 3 covers the results of [3]. Section 4 discusses the

subset-cover model (mostly the results of [6]), with the results of [4] appearing in Section 4.3.3. Traitor tracing is briefly discussed in Section 4.4.

2 Definitions

Denote by $\mathcal{U}$ the set of receivers and by $\mathcal{R} \subseteq \mathcal{U}$ the set of revoked receivers (users who we do not want to be able to decrypt the next transmission(s)). (We will write $\overline{\mathcal{R}}$ for $\mathcal{U} \setminus \mathcal{R}$, the set of privileged (*i.e.*, non-revoked) users.) Set $N := |\mathcal{U}|$ and $r := |\mathcal{R}|$. In practice, N will be an upper bound on the number of receivers allowed in the system. While we may broadcast assuming all N users are receiving, we need only be sure to associate a newly added user with an unused $u \in \mathcal{U}$ when they join the system.

We denote by $\mathcal{M}_F$ and $\mathcal{C}_F$ the message space (our goal is to securely broadcast messages in $\mathcal{M}_F$) and ciphertext space associated with an encryption algorithm F: $\mathcal{K} \times \mathcal{M}_F \to \mathcal{C}_F$ (with a key space $\mathcal{K}$). We assume such an F is given.

2.1 Defining Broadcast Encryption

For motivational purposes, we quickly examine a naïve approach to broadcast encryption: in this approach, the sender essentially encrypts the message one time for each privileged user. More precisely, the sender initially assigns and distributes (before any message transmissions) to each user $u \in \mathcal{U}$ a unique key $K_u \in \mathcal{K}$. To broadcast an $M \in \mathcal{M}_F$, the sender sends $B := (u_1, F(K_{u_1}, M)) \,||\, \cdots \,||\, \big(u_{N-r}, F\big(K_{u_{N-r}}, M\big)\big)$ to every user, where $(u_1, \ldots, u_{N-r}) = \overline{\mathcal{R}}$. To decrypt, a user $u \in \overline{\mathcal{R}}$ finds the pair beginning with u (say, (u, C)) in B and computes $F^{-1}(K_u, C) = M$. Notice that (assuming that F is secure) users can decrypt M if and only if they are in $\overline{\mathcal{R}}$ (a hybrid arguments shows that the probability that an adversary can break this scheme is at most $\frac{1}{N-r}$ times the probability that an adversary can break F). While this scheme requires each user to store a very small number of keys (1), the message length ($|B|$) is prohibitively large ($O(N-r)$).

In the sequel, we will rarely need the notation $\mathcal{M}_F$, $\mathcal{C}_F$, or F; we set them aside by instead solving the problem of getting a $K \in \mathcal{K}$ (a different one for each transmission[1]), rather than an arbitrary message $M \in \mathcal{M}_F$, to only the privileged users. Once the privileged users have K, we then send $F(K, M)$ to every user; the users who have this session key K can decrypt and get M and — again relying on the security of F — those who do not have K cannot get M. With these facts in mind, we make our most fundamental definition:

Definition 2.1. A *broadcast-encryption scheme* is a triple of algorithms (Setup, Broadcast, Decrypt) such that:

- The setup algorithm (Setup) takes a user ($u \in \mathcal{U}$) and constructs that receiver's private information $P_u \in \mathcal{P}$ (for some set $\mathcal{P}$ associated with the scheme).
- The broadcast algorithm (Broadcast) takes the list of revoked users $\mathcal{R}$ and the session key $K \in \mathcal{K}$ and outputs a broadcast message B (in, say, some message space $\mathcal{B}$).
- A user $u \in \mathcal{U}$ runs the decryption algorithm Decrypt(B, P_u, u) that will compute the K associated with B, assuming $u \in \overline{\mathcal{R}}$ (if not, Decrypt fails).

[1]Technically, we need only change keys when $\mathcal{R}$ changes or when the amount of ciphertext broadcast exceeds a security bound imposed by F.

In an effort to cut down the transmission size seen in the earlier naïve scheme, we can try the following (too-)simple scheme: SETUP picks a unique $K_S \in \mathcal{K}$ for every subset $S \subseteq \mathcal{U}$; SETUP then gives user u the concatenation $P_u := (S_1, K_{S_1}) \,||\, \cdots \,||\, \left(S_{2^{N-1}}, K_{S_{2^{N-1}}}\right)$, where $S_1, \ldots, S_{2^{N-1}}$ are the subsets of $\mathcal{U}$ that do *not* contain u. BROADCAST simply outputs a description of $\mathcal{R}$ (the session key K will be $K_{\mathcal{R}}$). Notice that only a user in $\overline{\mathcal{R}}$ will have $K = K_{\mathcal{R}}$. We have brought the message length ($|B|$) down to $\log r = O(\log N)$, but now the problem lies in the number of keys each user must keep: $O\left(2^N\right)$. (Throughout the paper, "log" means "logarithm base 2"; "ln" will be used (for the natural logarithm) at the end of Section 4.3.3.)

The primary motivation of broadcast-encryption research has been to find suitable middle ground between these two extremes. This paper will present various results that balance these two factors (number of keys per user vs. header length ($|B|$)).

2.2 Traitor Tracing

When an illegal decoder box is recovered, a server would like to know who constructed (*i.e.*, gave their private information P_u to help build) the box. Naor, Naor, and Lotspiech [6] present a tracing algorithm (which works in tandem with a large class of broadcast-encryption schemes) in which, though the identity of the culprits may not be discovered, any boxes using only keys from the culprits (including the captured box and the original, legitimate boxes given to the culprits) will be made to have a *functionality* of less than q (a parameter of the tracing algorithm), a term defined here:

Definition 2.2. For $p \in [0, 1]$, we say that a decoder box is *p-functional* (alternatively, the box has a *functionality of p*) if the box correctly decodes with probability at least p. The probability is taken over the randomness of the box, a uniformly chosen random message, and a uniform selection of $\overline{\mathcal{R}} \subseteq \mathcal{U}$ not containing any user involved in the construction of the box.

We also note that a tracing algorithm is not allowed to destroy the decryption functionality of a legitimate user.

The algorithm will work in the black-box model; that is, the algorithm's only interaction with the box will be feeding it ciphertexts and examining the results. We assume that the algorithm is resettable, *i.e.*, that we can return the box to its original state at any time.

In general, a tracing algorithm is evaluated based on (1) the decrease in efficiency it imposes on the broadcast-encryption scheme, (2) the size of coalitions against which it is resilient, and (3) the number of ciphertext queries required of the box. The algorithm in [6] has no negative effect on the broadcast-encryption scheme, it will be resilient against arbitrarily large coalitions, and it will require a quadratic number of ciphertext queries (up to log factors; in terms of the cover size and number of traitors) of the illegal box. More details on traitor tracing appear in Section 4.4.

3 Early Results

The first formal study of broadcast encryption appears in [3]. Several schemes (*e.g.*, [2]) predate the results of Fiat and Naor; however, these schemes suffer from one inadequacy or another. Some are "one-time" (the keys must be updated after every transmission) and some are not designed for an arbitrary setting. As such, we begin our discussion of broadcast-encryption schemes with a discussion of Fiat and Naor's paper.

3.1 Security

We note that the security definitions appearing in [6] (see Section 4.2) are more formal than those in [3] (and, thus, in this section).

Definition 3.1. For a subset $S \subseteq \mathcal{U}$, a broadcast-encryption scheme is said to be *resilient to a set* S if, for every $S' \subseteq \mathcal{U} \setminus S$, the secrets common to $u \in S'$ are "secure" against an adversary that is given all the secrets known by $u \in S$.

Fiat and Naor present two definitions for "secure":

Definition 3.2. A random variable σ is said to be *information-theoretically "secure"* against an adversary that is given κ if the distribution on σ equals the conditional distribution on σ given κ.

Definition 3.3. A random variable σ is said to be *computationally "secure"* against a computationally bounded (*e.g.*, probabilistic polynomial-time) adversary that is given κ if the adversary, given κ, cannot distinguish σ from a random value (distributed uniformly over the possible values for σ) with non-negligible probability.

When using the computational definition of security, the definitions of various types of resilience (Definitions 3.1, 3.4, and 3.5) require that the adversaries are computationally bounded.

Now, the key security definitions from [3]:

Definition 3.4. A broadcast-encryption scheme is said to be *k-resilient* if it is resilient to every set $S \subseteq \mathcal{U}$ of size k.

Definition 3.5. A broadcast-encryption scheme is said to be *(k, p)-resilient* if it is resilient to (at least) a $1 - p$ fraction of sets $S \subseteq \mathcal{U}$ of size k.

3.2 Summary of Broadcast-Encryption Examples from [3]

The results of [3] that we will discuss are summarized in Table 1 ("μ" and "D" are explained in Sections 3.4 and 3.4.1, respectively).

Remark 1. We assume that a description of $\mathcal{R}$ precedes each broadcast or is otherwise known to each user. As such, Table 1 does not account for this $\log r$ overhead in the header lengths ($|B|$). This is a reasonable decision, as all the schemes mentioned in this paper require the users to know $\mathcal{R}$ to be able to decrypt.

Notice that the table contains our first and second naïve results in rows (a) and (b″) (which will be seen to be a special case of row (b)), respectively. We will next present three (classes of) schemes in which Broadcast will send an empty message: (b), (c), and (d). Schemes (c) and (d) will be build upon (b). From these, the more complicated schemes (e) (and (e')) and (f) are built.

3.3 Zero-Message Schemes: (b), (c), and (d)

3.3.1 An Assumption-Free Scheme: (b)

Theorem 3.1. *There exists a (information-theoretically) k-resilient broadcast-encryption scheme that (1) requires each user to keep $\sum_{i=0}^{k} \binom{n-1}{i}$ keys and (2) has a header length of 0.*

	# keys/user	header length ($\|B\|$)	resilience	assumptions
(a)	1	$O(N-r)$	(any) k	NONE
(b)	$\sum_{i=0}^{k}\binom{N-1}{i}$	0	k	NONE
(b′)	N	0	1	NONE
(b″)	2^{N-1}	0	N	NONE
(c)	$\lceil \log N \rceil$	0	1	PRNGs exist
(d)	1	0	1	root extraction is hard
(e)	$O(k\mu \log N)$	$O(k^3 \log N)$	k	see (b′), (c), or (d)
(e′)	$O(k\mu(\log N)/(\log D))$	$O(k^3(\log N)D/(\log D))$	k	see (b′), (c), or (d)
(f)	$O(k\mu(\log k)(\log N))$	$O(k^2 \log N)$	k	see (b′), (c), or (d)

Table 1: Trade-offs in broadcast-encryption schemes found in [3]

Proof. We show here how to construct scheme (b), which will meet the requirements of the theorem.

In scheme (b), SETUP picks a unique $K_S \in \mathcal{K}$ for every subset $S \subseteq \mathcal{U}$ of size at most k; SETUP then gives user u the private information $P_u := \left\{\left(S_j, K_{S_j}\right) : |S_j| \leq k, u \notin S_j\right\}$. Notice that there are $\sum_{i=0}^{k}\binom{N-1}{i}$ pairs (S_j, K_{S_j}) in P_u. BROADCAST outputs the empty string. K will be $\bigoplus_{S \subseteq \mathcal{R}} K_S$, where the exclusive or is taken over S of size at most k. Notice first that a user $u \in \overline{\mathcal{R}}$ will know each of the K_S that are used to make K (and, thus, u will be able to compute K). Now notice that a coalition $S' \subseteq \mathcal{R}$ of at most k adversaries will not know $K_{S'}$ and, as $|S'| \leq k$, $K_{S'}$ is used to compute S, we conclude that S' will see K as random. Hence, (b) is (information-theoretically) secure against S', and, thus, is k-resilient. □

Notice that we can achieve the values in (b′) and (b″) by setting $k := 1$ and setting $k := N$, respectively.

3.3.2 A Scheme Based on One-Way Functions: (c)

In (b′), each user has N keys: we can lower this to $\lceil \log N \rceil$ by modifying the scheme; however, this scheme requires an additional assumption. For this scheme, we will need a (length-doubling) pseudo-random-number generator, which we will denote $f\colon \mathcal{K} \to \mathcal{K}^2$. Fiat and Naor point out that it is sufficient to assume the existence of one-way functions to guarantee the existence of such an f. More formally:

Theorem 3.2. *If one-way functions exist, then there exists a (computationally) k-resilient broadcast-encryption scheme that (1) requires each user to keep $\lceil \log N \rceil$ keys and (2) has a header length of 0.*

Proof. We show here how to construct scheme (c), which will meet the requirements of the theorem.

SETUP uses f to construct a tree of $2N-1$ keys (with N leaves); each leaf will represent a user in $\mathcal{U}$. (For notational convenience, we assume that N is a power of 2). We will assign a key to every node in the tree, denoting by K_S the key at the node whose descendant leaves are $S \subseteq \mathcal{U}$. SETUP will start with a seed key $K_{\mathcal{U}} \in \mathcal{K}$. K_{S_L} and K_{S_R} are constructed from the key K_S as $(K_{S_L}, K_{S_R}) := f(K_S)$, where S_L are the descendant leaves of the left subtree of S; similarly with S_R on the right. (We will see momentarily that we need not construct these keys explicitly; if the actual number of users is significantly fewer than N, we will certainly want to avoid constructing the whole tree.) To determine which $\log N$ keys user $u \in \mathcal{U}$ will get, simply imagine removing all the ancestors of u (including u; this makes for $1 + \log N$ nodes) from the tree: a forest of $\log N$

trees is created. SETUP simply sends u the key at the root of each of these $\log N$ trees. Notice that u, using the publicly known function f, can generate the key associated with every leaf node but 1 — the one associated with u.

As stated earlier, BROADCAST will do nothing. For a revoked user set $\mathcal{R}$, the key will be $K = \bigoplus_{u \in \mathcal{R}} K_{\{u\}}$. If the $K_{\{u\}}$ were all truly random, the security of this scheme would be just like that of scheme (b). But this is not a concern, as the $K_{\{u\}}$ can be seen to all be pseudo-random by a straightforward hybrid argument (the security of the broadcast scheme is $\frac{1}{\lceil \log N \rceil}$ times that of f). □

3.3.3 A Scheme Based on a Number-Theoretic Assumption: (d)

In scheme (c), we gave each user $\lceil \log N \rceil$ keys, requiring only the assumption that pseudo-random-number generators exist. We can reduce this number to 1 key per user by utilizing a different assumption: that root extraction modulo an RSA composite $A = PQ$ is hard. More formally:

Theorem 3.3. *If root extraction modulo an RSA composite is hard, then there exists a (computationally) k-resilient broadcast-encryption scheme that (1) requires each user to keep 1 key and (2) has a header length of 0.*

Proof. We show here how to construct scheme (d), which will meet the requirements of the theorem.

In scheme (d), every user $u \in \mathcal{U}$ is assigned a prime p_u such that any two primes p_u and p_v are relatively prime. (The assignment is public — *i.e.*, anyone can find p_u for any user $\mathcal{U}$). SETUP picks a secret $g \in \mathbb{Z}_A^\times$ with sufficiently large order. SETUP computes and sends user u its private information $P_u := \{g_u\}$ (where $g_u := g^{p_u}$).

Again, BROADCAST does nothing. For a revoked user set $\mathcal{R}$, the server computes the key $K := g^{\prod_{u \in \overline{\mathcal{R}}} p_u}$ (notice that $K \in \mathbb{Z}_A^\times$, so $\mathbb{Z}_A^\times \subseteq \mathcal{K}$; the keys are $\log A$ bits long). Notice that if a user u is in $\overline{\mathcal{R}}$, then they can compute $K = g_u^{\prod_{v \in \overline{\mathcal{R}} \setminus \{u\}} p_v}$.

To prove that the system is secure, we will show that user $u \in \mathcal{R}$ who can compute K can also compute g (contradicting our hardness assumption). Clearly u can compute $z := \prod_{v \in \overline{R}} p_v$. Notice that z and p_u are relatively prime, so we may construct $\alpha, \beta \in \mathbb{Z}_A^\times$ such that $\alpha z + \beta p_u = \gcd(z, p_u) = 1$. But this means that u can compute $K^\alpha {g_u}^\beta = g^{z\alpha} g^{p_u \beta} = g$, contradicting the hardness assumption. □

3.4 Schemes That Broadcast Messages: (e) and (f)

Fiat and Naor construct two k-resilient broadcast-encryption schemes (which we call (e) and (f)) from the 1-resilient schemes found in Section 3.3. In each of the following schemes, we assume the existence of a 1-resilient scheme that gives μ keys to each user (*i.e.*, if we use (b′), $\mu = N$; with (c), $\mu = \lceil \log N \rceil$; and with (d), $\mu = 1$). The number of keys each user will have in schemes (e) and (f) depends linearly on μ. Notice that (e) and (f) must assume the assumptions of the 1-resilient scheme they use (*e.g.*, the existence of one-way functions when using (c) or that root extraction is hard when using (d)).

3.4.1 A k-Resilient Scheme: (e)

Here we will have an optimization parameter $\ell \in \mathbb{N}$ (which we will optimize later); scheme (e) will have $\mu\ell$ keys. We will use another variable m (which we will also optimize later) constrained by the following: there must exist a family of functions $\{f_i\}_{i=1}^{\ell}$ (with $f_i \colon \mathcal{U} \to \{1, 2, \ldots, m\}$) such that, for

all $S \subseteq \mathcal{U}$ of size k, we are guaranteed at least one i such that $f_i|_S$ is injective. Fiat and Naor point out that these functions need not be explicitly constructed: the $f_i(u)$ that will be given to users (see Sections 3.4.1 and 3.4.2) can be chosen at random. The server can actually just generate them all from a pseudo-random-number generator and a seed. Alternatively, they could be generated by pairwise-independent functions .

The following three corollaries (of Theorem 3.7) present the main results for the three variants of scheme (e):

Corollary 3.4. *There exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O(kN \log N)$ keys and (2) has a header length of $O\big(k^3 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.7 and 3.1 (notice then that $\mu = N$). □

Corollary 3.5. *If one-way functions exist, then there exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O\big(k \log^2 N\big)$ keys and (2) has a header length of $O\big(k^3 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.7 and 3.2 (notice then that $\mu = \lceil \log N \rceil$). □

Corollary 3.6. *If root extraction modulo an RSA composite is hard, then there exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O(k \log N)$ keys and (2) has a header length of $O\big(k^3 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.7 and 3.3 (notice then that $\mu = 1$). □

Theorem 3.7. *If there exists a 1-resilient scheme that requires each user to hold μ keys and has a header length of 0, then there exists a k-resilient scheme that (1) requires users to keep $O(k\mu \log N)$ keys and (2) has a header length of $O(k^3 \log N)$.*

Proof. We show here how to construct scheme (e) from several 1-resilient schemes with the properties mentioned in the statement of the theorem. Scheme (e) will meet the requirements of the theorem.

Here we assume the existence of ℓm independent 1-resilient schemes that we will denote by $\textsc{Scheme}^{(i,j)} = \Big(\textsc{Broadcast}^{(i,j)}, \textsc{Setup}^{(i,j)}, \textsc{Decrypt}^{(i,j)}\Big)$ for $1 \le i \le \ell$ and $1 \le j \le m$. We will construct the k-resilient scheme (e) and call it $\textsc{Scheme} = (\textsc{Setup}, \textsc{Broadcast}, \textsc{Decrypt})$.

If $P_u^{(i,j)}$ denotes the output of the setup algorithm $\textsc{Setup}^{(i,j)}$ for user $u \in \mathcal{U}$, then scheme (e)'s $\textsc{Setup}$ gives user u the private information $P_u := \Big\{P_u^{(i,f_i(u))} \ : \ 1 \le i \le \ell\Big\}$.

Along with the revoked set $\mathcal{R}$, a randomly chosen key $K \in \mathcal{K}$ is given to $\textsc{Broadcast}$, which then picks random $K_1, K_2, \ldots, K_{\ell-1} \in \mathcal{K}$ and sets $K_\ell := K \oplus \Big(\bigoplus_{i=1}^{\ell-1} K_i\Big)$. Denote by $B^{(i,j)}$ the output of $\textsc{Broadcast}^{(i,j)}\big(\mathcal{R}^{(i,j)}, K_i\big)$, where $\mathcal{R}^{(i,j)} := \mathcal{R} \cup \overline{f_i^{-1}(j)} = \mathcal{R} \cup (\{u \in \mathcal{U} \ : \ f_i(u) \ne j\})$. Then $\textsc{Broadcast}$ outputs the concatenation of all ℓm of the $B^{(i,j)}$s.

To see that any $u \in \overline{\mathcal{R}}$ can construct K, notice that (for each i) $u \notin \mathcal{R}^{(i,f_i(u))}$, so user u can construct $K_i = \textsc{Decrypt}^{(i,f_i(u))}\Big(B^{(i,f_i(u))}, P_u^{(i,f_i(u))}, u\Big)$. Then, u can construct $K = \bigoplus_{i=1}^{\ell} K_i$. Notice that each user has $\mu\ell$ keys and that the message length ($|B|$) equals $\mu\ell m$.

To see that K appears random (or pseudo-random, if the 1-resilient $\textsc{Scheme}^{(i,j)}$s are only computationally secure) to an arbitrary coalition $S \subseteq \mathcal{R}$ of revoked users of size at most k, fix

an i^* such that $f_{i^*}|_S$ is injective. Since $K = \bigoplus_{i=1}^{\ell} K_i$, it is enough to show that K_{i^*} appears (pseudo-)random to S (in the pseudo-random case, that K is pseudo-random will from a hybrid argument).

Notice that, since the 1-resilient schemes are independent, K_{i^*} appears random to S, given each of the messages $B^{(i,j)}$ (for $i \neq i^*$). We will now show, for each j, that K_{i^*} stills appears (pseudo-)random to S, even knowing $B^{(i^*,j)}$. Thus, K_{i^*} appears (pseudo-)random to S, given *all* the $B^{(i^*,j)}$ (if the 1-resilient schemes are pseudo-random, this follows from a hybrid argument).

Fix $j \in \{1, 2, \ldots m\}$. Since $f_{i^*}|_S$ is injective, there is at most one $u^* \in S$ such that $f_{i^*}(u^*) = j$. (For convenience, we assume the worst, that such a u^* exists.) Recall that a user $u \in \mathcal{U}$ has only the keys for schemes $\text{SCHEME}^{(i,f_i(u))}$ (for all i) and, thus, the only user in S that has keys for $\text{SCHEME}^{(i^*,j)}$ is user u^*. Hence, in $\text{SCHEME}^{(i^*,j)}$, S has the keys for at most one user and, thanks to the 1-resilience of $\text{SCHEME}^{(i^*,j)}$, K_{i^*} appears (pseudo-)random to S.

To optimize m and ℓ, fix $S \subseteq \mathcal{U}$ of size at most k, and notice that when $m \geq 2k^2$, for each i, the probability that a randomly constructed $f_i|_S$ is injective is at least $1 - \binom{|S|}{2}/m \geq 3/4$. So we set $m := 2k^2$. Now notice that if we set $\ell := k \log N$, then the probability that *no* $f_i|_S$ is injective is at most $(1/4)^\ell = 1/N^{2k}$. Thus, the probability that, for each $S \subseteq \mathcal{U}$ of size k, at least one of the $f_i|_S$ is injective, is at least $1 - \binom{N}{k}/N^{2k} \geq 1 - 1/N^k$. □

Remark 2. To realize scheme (e′), simply choose $m := Dk^2$ and $\ell := \frac{2}{\log(2D)}(k \log N)$. Then $1 - \binom{|S|}{2}/m \geq 1 - \frac{1}{2D}$, so the probability that *no* $f_i|_S$ is injective is at most $\left(\frac{1}{2D}\right)^\ell$. Hence, the $\{f_i\}$ function family is as required with probability at least $1 - \binom{N}{k}\left(\frac{1}{2D}\right)^\ell \geq 1 - \frac{N^k}{(2D)^\ell} \geq 1 - \frac{1}{N^k}$.

To achieve (k, p)-resiliency, take $m = k^2$ and $\ell = \log(1/p)$. Notice that this has the effect making the number of keys per user $\mu\ell = \mu \log(1/p)$ and the message length $\mu\ell m = \mu k^2 \log(1/p)$.

3.4.2 A Better k-Resilient Scheme: (f)

Here we give three corollaries (of Theorem 3.7) that present the results for the three variants of scheme (f):

Corollary 3.8. *There exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O(kN(\log k)(\log N))$ keys and (2) has a header length of $O\big(k^2 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.11 and 3.1 (notice then that $\mu = N$). □

Corollary 3.9. *If one-way functions exist, then there exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O\big(k(\log k)\big(\log^2 N\big)\big)$ keys and (2) has a header length of $O\big(k^2 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.11 and 3.2 (notice then that $\mu = \lceil \log N \rceil$). □

Corollary 3.10. *If root extraction modulo an RSA composite is hard, then there exists a k-resilient broadcast-encryption scheme that (1) requires each user to keep $O(k(\log k)(\log N))$ keys and (2) has a header length of $O\big(k^2 \log N\big)$.*

Proof. The theorem follows immediately from combining Theorems 3.11 and 3.3 (notice then that $\mu = 1$). □

Theorem 3.11. *If there exists a 1-resilient scheme that requires each user to hold μ keys and has a header length of 0, then there exists a k-resilient scheme that (1) requires users to keep $O(k\mu(\log k)(\log N))$ keys and (2) has a header length of $O\big(k^2 \log N\big)$.*

Scheme (f) is constructed from several 1-resilient schemes with the properties mentioned in the statement of the theorem. Scheme (f) meets the requirements of the theorem. It is a multi-level generalization of scheme (e); however, there is little new in this result. Hence, we skip the details (which, of course, can be found in [3]).

4 The Subset-Cover Model

In [6], Naor, Naor, and Lotspiech present and analyze an abstract framework for broadcast schemes. Their so-called *subset-cover* framework abstracts several previously known schemes as well as two new schemes that they also present. As opposed to the schemes of [3], these schemes are resilient to any size coalition of revoked users (even all r of them). ([6] calls a scheme with this property *r-flexible*.) Another important improvement seen in [6] is in, of course, the number of keys per user and the header length required by the new schemes — their scheme (h) was the best known broadcast scheme when it was published. Table 2 details these two quantities for the schemes covered in [6] (we discuss its schemes (g) and (h) in Sections 4.3.1 and 4.3.2, respectively) and [4] (see Section 4.3.3). In Section 4.4, we discuss a seamlessly integrated traitor-tracing algorithm for broadcast schemes in the subset-cover framework that satisfy a particular property.

	name	# keys/user	header length ($\lvert B\rvert$)
(g)	complete subtree	$O(\log N)$	$r \log(N/r)$
(h)	subtree difference	$O(\log^2 N)$	$2r - 1$
(i)	basic LSD	$O\left(\log^{3/2} N\right)$	$4r - 2$
(j)	general LSD	$O\left(\frac{1}{\varepsilon} \log^{1+\varepsilon} N\right)$	$\frac{1}{\varepsilon}(2r - 1)$
(j′)	optimal LSD	$O(\log N \log\log N)$	$O(r \log\log N)$
(k)	inclusion-exclusion trees	see (h),(i),(j), or (j′)	$O(t)$ (t = # conditions)

Table 2: Trade-offs in broadcast-encryption schemes found in [6] ((g)–(h)) and [4] ((i)–(k))

4.1 Definitions

Broadcast-encryption schemes in the subset-cover framework (henceforth, *subset-cover schemes*) use an auxiliary scheme (*e.g.*, a block cipher) to aid in the delivery of K to the privileged users; we will denote its key space by $\mathcal{L}$, its ciphertext space by $\mathcal{C}$ (the message space will be $\mathcal{K}$) and the actual cipher by E: $\mathcal{L} \times \mathcal{K} \to \mathcal{C}$.

A subset-cover scheme is a broadcast-encryption scheme with the following additional properties:

Setup will pick subsets $S_1, S_2, \ldots, S_w \subseteq \mathcal{U}$ and assign each S_i a long-lived key $L_i \in \mathcal{L}$. If the L_i are chosen randomly and independently, we will have information-theoretic security; otherwise, the L_i will be chosen as a function of some other (secret) information, and we have computational security. Setup gives a user $u \in \mathcal{U}$ private information P_u that is sufficient for u to compute L_i for every i such that $u \in S_i$.

Broadcast chooses a random $K \in \mathcal{K}$. Given a list of revoked users $\mathcal{R}$, Broadcast calls an algorithm Cover($\overline{\mathcal{R}}$) to partition the set of privileged users into $\{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$. Then, Broadcast sends the message header $B = (i_1, E(L_{i_1}, K)) \,||\, \cdots \,||\, (i_m, E(L_{i_m}, K))$ to each user.

DECRYPT will be given a header $B = (i_1, C_1) \,||\, \cdots \,||\, (i_m, C_m)$, a user $u \in \overline{\mathcal{R}}$, and that user's private information P_u. DECRYPT calls an algorithm SUBSET$\left(u, \{S_{i_j}\}_{j=1}^{m}\right)$ to find an j such that $u \in S_{i_j}$. Then, DECRYPT calls an algorithm L-KEY(P_u, u, i_j) to construct L_{i_j}. DECRYPT then outputs $K = E^{-1}(L_{i_j}, C_{i_j})$.

Thus, a subset-cover scheme can be described by (1) the collection of subsets $\{S_i\}_{i=1}^{w}$, (2) the key construction for each member of $\{L_i\}_{i=1}^{w}$, (3) its COVER algorithm, (4) its SUBSET algorithm, and (5) its L-KEY algorithm.

Remark 3. The algorithms E and F are used in different manners: it is easy to change F's keys (in fact, F will generally never be called with the same key twice), but E's keys (the $L_i \in \mathcal{L}$) are long-lived. Care in the selection of E (and the size of $\mathcal{L}$) should take this into account. On the other hand, that F's keys are short-lived can be taken advantage of in allowing for a smaller $\mathcal{K}$ as well as allowing the use a less-secure F.

4.2 Security

We begin our discussion of subset-cover security with several definitions. We then finish with a theorem that describes sufficient conditions for a subset-cover scheme to be secure.

Definition 4.1. A subset-cover scheme is said to be *secure* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible:

- $\mathcal{A}$ sets $\mathcal{R} := \emptyset$.
- $\mathcal{A}$ repeats the following (at most $N - 1$ times):
  - $\mathcal{A}$ adaptively sends an arbitrary number of $(\mathcal{R}_i, K_i)$ queries (where $\mathcal{R}_i \subseteq \mathcal{U}$ and $K_i \in \mathcal{K}$) and receives $B_i = \text{BROADCAST}(\mathcal{R}_i, K_i)$; during these adaptive queries, $\mathcal{A}$ may ask for the result of DECRYPT(B, P_u, u) for as many B and $u \in \overline{\mathcal{R}}$ as it wants.
  - $\mathcal{A}$ adds a new u to $\mathcal{R}$
  - SETUP sends $\mathcal{A}$ the keys P_u.
- $\mathcal{A}$ chooses and sends $K \in \mathcal{K} \setminus \{K_i\}_i$ (a key that has not been used previously).
- $K_R \in \mathcal{K}$ is picked at random and a coin is flipped; then, $\mathcal{A}$ is sent either BROADCAST$(\mathcal{R}, K)$ or BROADCAST$(\mathcal{R}, K_R)$.
- $\mathcal{A}$ wins if it can guess which of K and K_R was used.

Definition 4.2. A subset-cover scheme is said to be *key-indistinguishable* if, for any probabilistic polynomial-time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible:

- $\mathcal{A}$ picks $i \in \{1, 2, \ldots, w\}$.
- SETUP sends $\mathcal{A}$ the secret keys $\left\{P_u \,:\, u \in \overline{S_i}\right\}$
- $L_R \in \mathcal{L}$ is picked at random and a coin is flipped; then, $\mathcal{A}$ is sent either L_i or L_R.
- $\mathcal{A}$ wins if it can guess which of L and L_R was given.

Now, the main security result:

Theorem 4.1. *If a key-indistinguishable subset-cover algorithm uses (chosen-plaintext-)secure E and F, then it is secure.*

4.3 Three Subset-Cover Examples

All three examples presented here (the first two from [6], the last — technically several instances of the same general example — from [4]) share several properties. All view the users $\mathcal{U}$ as N leaves on a tree (with $2N-1$ nodes (including the leaves)). We will assume that N is a power of 2, since we can simply round up any other N and suffer only minute penalties (our results depend on $\log N$ in at most a quadratic manner). We will use the following notation: $\mathfrak{T}(\mathcal{R})$ will denote the Steiner tree induced by the root and the leaves that represent the members of $\mathcal{R}$. For any tree $\mathfrak{T}$, we denote by $\mathfrak{B}(\mathfrak{T})$ the tree's external border, *i.e.*, the nodes in the original tree of $2N-1$ nodes that are children of nodes in $\mathfrak{T}$ but are *not* in $\mathfrak{T}$ themselves.

4.3.1 The Complete-Subtree Method: Scheme (g)

Theorem 4.2. *There exists a secure broadcast-encryption scheme that (1) requires each user to keep $1+\log N$ keys and (2) has a header length (for $r>0$) of at most $r\log(N/r)$ (recall that $r=|\mathcal{R}|$). (If $r=0$, the header length is 1.)*

Proof. We show here how to construct scheme (g), which will meet the requirements of the theorem.

The subsets $\{S_i\}_{i=1}^{w}$ ($w=2N-1$) are constructed so that each node (ν_i) in the tree is associated with exactly one subset: the subset containing all the leaves that descend from ν_i. Notice that each $u\in\mathcal{U}$ is in exactly $1+\log N$ subsets (the subsets associated with the ancestors of u).

The keys associated with each subset are simply randomly and independently chosen $L_i\in\mathcal{L}$.

The Cover algorithm takes a group of privileged users $\overline{\mathcal{R}}$ and constructs the cover $\{S_{i_j}\}_{j=1}^{m}$, where the S_{i_j} are exactly the subsets associated with the nodes $\nu_{i_j}\in\mathfrak{B}:=\mathfrak{B}(\mathfrak{T}(\mathcal{R}))$.

Notice that the cover size m equals the number of degree-1 nodes in $\mathfrak{T}$, so, using Lemma 4.3, we conclude that $m\le r\log(N/r)$.

To find which subset a user $u\in\overline{\mathcal{R}}$ is in, the Subset algorithm follows the path from the root of the tree to the node representing u until it first hits a node $\nu_{i_j}\in\mathfrak{B}$ (which it will hit, because $u\in\overline{\mathcal{R}}$). Notice that the S_{i_j} associated with ν_{i_j} is in the cover, and that $u\in S_{i_j}$, so L-Key will find L_{i_j} by searching P_u.

As the L_i are chosen independently and $\mathcal{R}\cap\bigcup_{j=1}^{m}S_{i_j}=\emptyset$, this scheme is key-indistinguishable and thus (by Theorem 4.1) secure. □

Now we prove the lemma that was needed by Theorem 4.2:

Lemma 4.3. *For non-empty $\mathcal{R}$, the number of degree-1 nodes in $\mathfrak{T}(\mathcal{R})$ is at most $r\log(N/r)$.*

Proof. Our proof proceeds by induction on the depth of the tree ($\log N$). The 1-node tree ($\log N=0$) has zero edges, as do any subtrees, so the number of degree-1 nodes in $\mathfrak{T}(\mathcal{R})$ must be 0. Now we assume that the lemma holds when the tree is of depth i (when $N=2^i$) and show it holds when the tree is of depth $i+1$ (when $N=2^{i+1}$).

If all the nodes representing $\mathcal{R}$ descend from one particular child of the root, notice that we may apply the inductive case immediately to the left or right subtree of the root. Thus, the number of degree-1 nodes in $\mathfrak{T}$ is the sum of 1 (from the root) plus at most $r\log(2^i/r)$, which is in turn no more than $r+r\log(2^i/r)=r\log(N/r)$.

Otherwise, notice that the number of degree-1 nodes in the depth-i tree is the number of degree-1 nodes in the left subtree plus the number of degree-1 nodes in the right subtree. Denote by r_L the number of nodes representing the members of $\mathcal{R}$ that descend from the left subtree of the root (*i.e.*, $r_L:=|\mathcal{R}\cap S_{i_L}|$, where ν_{i_L} is the left child of the root). Similarly, write $r_R:=|\mathcal{R}\cap S_{i_R}|$ (so

$r = r_L + r_R$). Then, using the inductive case twice, we see that the number of degree-1 nodes in $\mathfrak{T}$ is at most $r_1 \log\left(2^i/r_1\right) + r_2 \log\left(2^i/r_2\right)$. Since $r_1 \log r_1 + r_2 \log r_2 + r \geq r \log r$ (whenever r_1 and r_2 are positive integers), we see that the number of degree-1 nodes in $\mathfrak{T}$ is at most $r \log\left(2^{i+1}/r\right)$. □

4.3.2 The Subtree-Difference Method: Scheme (h)

We note that [6] refers to this as the "sub*set*-difference method"; we feel that using "sub*tree*" better represents the algorithm.

Theorem 4.4. *There exists a secure broadcast-encryption scheme that (1) requires each user to keep* $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$ *keys and (2) has a header length of at most* $2r - 1$.

Proof. We show here how to construct scheme (h), which will meet the requirements of the theorem.

The subsets are constructed so that each pair of nodes (ν_i, ν_j), where ν_i is a proper ancestor of ν_j, is associated with exactly one subset: the subset containing all the leaves that descend from ν_i and do *not* descend from ν_j (*i.e.*, the difference of the subtrees rooted at ν_i and ν_j). For notational convenience, we will denote the subsets in this scheme by $\{S_{i,j}\}$, where a particular subset $S_{i,j}$ is the subset associated with the pair of nodes (ν_i, ν_j).

Since each $u \in \mathcal{U}$ is in $\Theta(N)$ of these subsets, we cannot simply assign a unique key to each subset and then send u the keys for the subsets that u belongs to. To construct the keys $L_{i,j} \in \mathcal{L}$, we will require a length-tripling pseudo-random-number generator $f\colon \mathcal{L} \to \mathcal{L}^3$. (For notational convenience, break the output of f into f_A, f_B and f_C, so $f(L) = f_A(L)||f_B(L)||f_C(L)$.) We will utilize an intermediate construction: meta-keys $L^*_{i,j}$, for all pair (ν_i, ν_j) where ν_i is an ancestor of ν_j (*including* $\nu_i = \nu_j$). To construct the meta-key $L^*_{i,j}$, we first randomly and independently choose $L^*_{i,i} \in \mathcal{L}$ (one for each node ν_i). The remaining meta-keys are defined recursively: for ν_x that is a left child of ν_j, $L^*_{i,x} := f_A\left(L^*_{i,j}\right)$; if ν_x is a right child, $L^*_{i,x} := f_B\left(L^*_{i,j}\right)$. The actual key associated with subset $S_{i,j}$ will then be $L_{i,j} := f_C\left(L^*_{i,j}\right)$.

We, however, do *not* give a user $u \in \mathcal{U}$ all the $L_{i,j}$ for the $S_{i,j}$ it belongs to (again, there are too many); we just give it enough information to be able to compute any of those $L_{i,j}$ on its own. SETUP sends to user u exactly the meta-keys $L^*_{i,j}$ associated with every $S_{i,j}$ such that (ν_i, ν_j) is a proper-ancestor-descendant pair and the node representing u is a descendant of both ν_i and the *sibling* of ν_j. Additionally, each user receives a special key (which we denote $L_{\mathcal{U}}$) for when $\mathcal{R} = \emptyset$.

To count the number of meta-keys a user $u \in \mathcal{U}$ will receive, notice that for a ν_i that is y nodes above the node representing u, there will be y keys of the form $L^*_{i,j}$ in P_u. Since (for a given u) there will be one such ν_i for each y ranging from 0 to $\log N$, the number of keys each user receives is (adding 1 for $L_{\mathcal{U}}$) $|P_u| = 1 + \sum_{y=0}^{\log N} y = \frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$.

The COVER algorithm will take a privileged set $\overline{\mathcal{R}}$ and build the partition $\{S_{i_x,j_x}\}_{x=1}^{m}$ iteratively. Begin by setting $\mathfrak{T}^* := \mathfrak{T}(\mathcal{R})$ and continue the following shrinking process until $\mathfrak{T}^*$ is a chain:

> Pick two leaves, ν_c and ν_d, such that their nearest common ancestor (call it ν_0) has only two degree-1 descendants (ν_c and ν_d). Call the child of ν_0 that is ν_c's ancestor (possibly ν_c itself) ν_a. Similarly construct ν_b from ν_d. If $\nu_a \neq \nu_c$, add $S_{a,c}$ to the cover; similarly, if $\nu_b \neq \nu_d$, add $S_{b,d}$ to the cover. Remove all proper descendants of ν_0 from $\mathfrak{T}^*$.

If the resulting chain has more than 1 node, add a final subset $S_{a,c}$ to the cover, where ν_a and ν_c are the two ends of the chain.

To see that the header length is at most $2r - 1$, notice that in each iteration above, the number of leaves in $\mathfrak{T}^*$ decreases by 1, starting at r and ending at 1. During each iteration, at most 2

subsets are added; hence, the iterative process contributes at most $2(r-1)$ subsets to the cover. Finally, one more subset may by added when examining the final chain, for a total of at most $2(r-1)+1 = 2r-1$ subsets.

For a given $u \in \overline{\mathcal{R}}$, the SUBSET algorithm finds a subset (in the cover) that contains u by finding the nearest ancestor of the node representing u (call this ν_i) such that $S_{i,j}$ is in the cover (for some j).

To get $L_{i,j}$, we find the nearest common ancestor of ν_j and the node representing u and call it ν_a. Let ν_b be the child of ν_a that is *not* an ancestor of u; notice that $L^*_{i,b} \in P_u$. Using f_A and f_B appropriately, we can compute $L^*_{i,j}$ from $L^*_{i,b}$. Finally, recall that $L_{i,j} = f_C\left(L^*_{i,j}\right)$.

We now see why this scheme is key-indistinguishable and, thus, (by Theorem 4.1) secure. Assume it is not key-indistinguishable. So, an adversary $\mathcal{A}$ that picks (i,j) will be able to distinguish $L_{i,j}$ from $L_R \in \mathcal{R}$ (chosen at random), given all the keys $\left\{P_u \ : \ u \in \overline{S_{i,j}}\right\}$. Notice that, for $u \notin S_i$ (the descendants of ν_i), $L_{i,j}$ is information-theoretically independent of P_u, so any $\mathcal{A}$ must be able to distinguish $L_{i,j}$ from L_R given only P_u for $u \in \overline{S_{i,j}} \cap S_i = S_j$. Denote by ν_a the child of ν_i that is *not* an ancestor of ν_j. Denote by $\nu_{b_1}, \ldots, \nu_{b_z}$ the siblings of nodes on the path from ν_a to ν_j. Denote by ν_c and ν_d the children of ν_j (if it is not a leaf). Now notice that all of the $L^*_{i,j} \in \bigcup_{u \in S_j} P_u$ can be generated (by applications of f_A and f_B) from $L^*_{b_1,b_1}, \ldots, L^*_{b_z,b_z}$ and (if ν_j is not a leaf) $L^*_{c,c}$ and $L^*_{d,d}$. Notice that these $z+2 \leq (\log N - 1) + 2$ labels were generated independently, so, via a hybrid argument, we see that $\mathcal{A}$ must be able to distinguish the output of f from random (the security falls from that of f by a factor of $\frac{1}{(\log N+1)}$), contradicting that f is a pseudo-random-number generator. □

4.3.3 The Layered Subtree-Difference Method(s): Schemes (i), (j), and (k)

We note that [4] refers to this as the "layered-sub*set*-difference method"; as in Section 4.3.2, we feel that using "sub*tree*" better represents the algorithm.

In Section 4.3.1, we saw a way to give each user $O(\log N)$ keys, but we were required to send a header of length $O(r \log(N/r))$. Then we saw an improvement in Section 4.3.2 to a header of length $O(r)$ at the expense of increase the number of keys per user to $O\left(\log^2 N\right)$. Halevy and Shamir [4] present an algorithm that is nearly the best of both worlds: while retaining a header of length $O(r)$ (with a slightly larger constant), they are able to achieve extremely close to $O(\log N)$ keys per user.

To get there, we initially present a practical scheme (Theorem 4.6) that requires $O\left(\log^{3/2} N\right)$ keys per user (and $|B| = O(r)$). Afterwards, we discuss the more-theoretical scheme (Theorem 4.7) that gets essentially $O(\log N)$ keys per user.

Before seeing the initial scheme, we present some preliminaries. For notational convenience, we both define $q := \sqrt{\log N}$ and assume it is an integer. We return to the tree and sets $S_{i,j}$ of Section 4.3.2. For any node ν in the tree, we define $D(\nu) \in \{0, 1, \ldots, \log N\}$ to be the depth of ν (the root has depth 0 and the leaves have depth $\log N$).

Definition 4.3. A non-empty set $S_{i,j}$ (as defined in Section 4.3.2) is called *useful* if $q | D(\nu_i)$ or $qa < D(\nu_i) < D(\nu_j) \leq q(a+1)$ (for some $a \in \mathbb{Z}_q$).

This definition is motivated by the following idea: each user is the member of too many of the $S_{i,j}$s (*i.e.*, each user receives too many keys). The solution offered by the layered-subtree-difference (LSD) scheme is to define a small enough subcollection of the $S_{i,j}$s as *useful*, and only use these for our covers (and, consequently, for deciding which keys to initially distribute to users). The following lemma helps to clarify why the LSD scheme is extremely similar to the subtree-difference

scheme and, thus, requires little additional explanation (beyond proving the claimed bounds on the number of keys per user and $|B|$):

Lemma 4.5. *Every non-empty $S_{i,j}$ is either useful or the union of two useful sets.*

Proof. Let $S_{i,j}$ be a non-empty set. If $S_{i,j}$ is useful, we are done. Otherwise, pick $a \in \mathbb{Z}_q$ such that $qa < D(\nu_i) < q(a+1)$. Since $S_{i,j}$ is not useful, $q(a+1) < D(\nu_j)$. Pick k so that ν_k is the ancestor of j at depth $q(a+1)$. Notice that both $S_{i,k}$ and $S_{k,j}$ are useful. To complete the proof, realize that $S_{i,j} = S_{i,k} \cup S_{k,j}$. □

Theorem 4.6. *There exists a secure broadcast-encryption scheme that (1) requires each user to keep $\log^{3/2} N$ keys and (2) has a header length of at most $4r - 2$.*

Proof. We show here how to construct scheme (i), which will meet the requirements of the theorem.

Scheme (i) uses the useful subsets of scheme (h) as its collection of subsets. The $L_{i,j}$ (and $L^*_{i,j}$) are constructed in exactly the same fashion as in scheme (h). Scheme (i) executes the same COVER algorithm as scheme (h), replacing $S_{i,j}$ that are not useful with two useful sets $S_{i,k}$ and $S_{k,j}$ guaranteed by Lemma 4.5. For a user to determine which subset it is in (for a particular $\mathcal{R}$), it uses the same SUBSET algorithm as in scheme (h): find the nearest ancestor of u (call this ν_i) such that $S_{i,j}$ is in the cover (for some j). (Notice that if u would have found $S_{a,b}$ when running SUBSET in scheme (h), either $S_{a,b} = S_{i,j}$ or $S_{a,b} = S_{c,i} \cup S_{i,j}$ for some c.) The L-KEY algorithm for a user to compute its $L_{i,j}$ is the same as in scheme (h).

To complete the proof, we need only show that each user keeps $\log^{3/2} N$ keys and that the header length is at most $4r - 2$. The latter is easy to see: if scheme (h) would cover a particular $\overline{\mathcal{R}}$ with m' sets, scheme (i) will require no more than $2m'$ (it can do no worse than the case when every set is not useful and must be split). But Theorem 4.4 shows that $m' \leq 2r - 1$, so we see that the header length in scheme (i) is at most $4r - 2$.

To count the number of keys that each user has, first notice that for a user $u \in \mathcal{U}$, the set $S_{i,j} \in P_u$ (in scheme (i)) if and only if $S_{i,j}$ is in scheme (h)'s P_u *and* $S_{i,j}$ is useful. Hence, we need to count the number of (ν_i, ν_j) pairs such that the node associated with u is a descendant of both ν_i and the sibling of ν_j *and* $S_{i,j}$ is useful.

We will count the two disjoint types of useful sets separately. First, the sets where $q|D(\nu_i)$. Fix i and set $a := D(\nu_i)/q$. Notice that there are $q^2 - qa$ levels below ν_i and, as there is only one ν_j per level (for a fixed ν_i), there are exactly $q^2 - qa$ distinct $S_{i,j}$ for this ν_i. For each $a \in \{0, 1, \ldots, q\}$, there is at depth qa exactly one ancestor of the node associated with u. Thus, there are $\sum_{a=0}^{q} \left(q^2 - qa\right) = \frac{1}{2}q^2(q+1)$ total $S_{i,j}$s sent to u that are the first kind of useful.

We finish our calculation by examining (i, j) such that $qa < D(\nu_i) < D(\nu_j) \leq q(a+1)$ (for some $a \in \mathbb{Z}_q$). Fix an $a \in \mathbb{Z}_q$ and set $\Delta := D(\nu_i) - qa \in \{1, 2, \ldots, q-1\}$. Notice that each $\Delta \in \{1, 2, \ldots, q-1\}$ is associated with exactly one ν_i. Now notice that there are $q - \Delta$ possible ν_j for this ν_i and, thus, $\sum_{a=0}^{q-1} \sum_{\Delta=1}^{q-1} (q - \Delta) = \frac{1}{2}q^2(q-1)$ total $S_{i,j}$s sent to u that are the first kind of useful.

Totaling our two (disjoint) counts, we see that each user receives $\frac{1}{2}q^2(q+1) + \frac{1}{2}q^2(q-1) = q^3 = \log^{3/2} N$ keys. □

Remark 4. The choice of $q = \sqrt{\log N}$ was optimal. If useful sets were defined so $x|D(\nu_i)$ or $xa < D(\nu_i) < D(\nu_j) \leq q(a+1)$ (for $a \in \mathbb{Z}_y$, where $xy = \log N$), then the total number of keys sent to a user would be

$$\left(\sum_{a=0}^{y}(xy - xk)\right) + \left(\sum_{a=0}^{y-1}\sum_{\Delta=1}^{x}(x - \Delta)\right) = \frac{1}{2}xy(x+y) = \frac{1}{2}(\log N)\left(x + \frac{\log N}{x}\right) ,$$

which is maximized at $x = \sqrt{\log N}$.

The generalization of the LSD scheme is based on realizing that, if we allow the $S_{i,j}$ to be split into $d > 2$ sets, we can reduce even further the number of keys required by each user (this also increases our upper bound on the header size $|B|$ to $d(2r-1)$).

Halevy and Shamir then suggest an abstraction to help generalize LSD. First, notice that the subtree-difference scheme (scheme (h)) and the basic implementation of LSD (scheme (i)) broadcast (*i.e.*, construct their header B) using the same atomic operation: send a message that can be decrypted by exactly the members of $S_{i,j}$ (they then repeat this over various (i,j)). Now view the tree (with leaves representing $\mathcal{U}$) as directed graph first with no additional edges. View taking a transition from ν to ν' as sending a message to the descendants of the sibling of ν'. Let $\nu_{c_0} = \nu_i, \nu_{c_1}, \ldots, \nu_{c_z} = \nu_j$ be the shortest path from ν_i to ν_j (in this case, clearly the only path). Notice that taking this path corresponds to exactly the z messages that the complete-subtree scheme (scheme (g)) would need to use to send a message to the members of $S_{i,j}$. Notice also that the number of keys each user knows is equal to the number of edges that are part of a path from the root to the node representing that user.

Now add the transitive closure of the transitions and notice that we have a graph representing scheme (h). There are $O(\log^2 N)$ edges on a path from the root to any leaf (which corresponds to each user having $O(\log^2 N)$ keys), but one can go from any ν_i to any ν_j in 1 step (the members of each $S_{i,j}$ can be covered by 1 subset). The basic LSD scheme adds only some of the transitive closure to the original graph — the edges corresponding to the useful sets — and guarantees a path of at most 2 edges from any i to any j.

Fix a leaf; we now focus only on solving the problem for the path from the root to that leaf. We will supplement the original edges with only some of the transitive closure (for a fixed d, we will end up adding $O\left(\log^{1+\frac{1}{d}} N\right)$ edges) — enough to guarantee a path from any ν_i to any descendent ν_j with at most d edges. In order to be able to use scheme (h) with only minor modifications as in scheme (i) (where we use scheme (h) and, whenever an $S_{i,j}$ that we do not have is needed, we simply substitute at most d smaller sets), we must impose one further condition on the edges. Recall that in P_u are the meta-keys for siblings of the nodes on the path from ν_i to ν_j (except ν_i's sibling), for each $S_{i,j}$ containing u. To have these at our disposal, we must add this condition (called the *shrinkage* condition by [4]): if there is an edge from ν_i to ν_j, there must be an edge from ν_i to every descendant ν_k that is an ancestor of ν_j. Before giving a formal statement of the result, we present an analogue to the already-discussed concept of useful:

Definition 4.4. For $z \in \{0, 1, \ldots, d-1\}$, a node ν_i is called *z-useful* if $b^z | i$, but $b^{z+1} \nmid i$. (We also say that such a node has *utility* of z.)

In the $d = 2$ (basic) LSD scheme, $S_{i,j}$ was a useful set if i was 0-useful and $j > i$ was at most the next multiple of q (beyond i) or if i was 1-useful and $j > i$ was at most the next multiple of q^2 (*i.e.*, any value greater than i). In the following theorem, we will present a scheme that takes the natural extension, making $q = \log^{1/d} N$, and using the sets $S_{i,j}$ where, for each i, j ranges from $i+1$ to the next multiple of q^z (where z is the utility of i).

Theorem 4.7. *There exists a secure broadcast-encryption scheme that (1) requires each user to keep fewer than $\frac{1}{2} d \log^{1+1/d} N$ keys and (2) has a header length of at most $d(2r-1)$.*

Proof. We show here how to construct (at most) $\frac{1}{2} d \log^{1+1/d} N$ edges (all directed down and satisfying the shrinkage condition) on the nodes from the root to a leaf such that there is a length d (or

less) path from any node ν_i to any descendant ν_j. This implies how to construct scheme (j), which will meet the requirements of the theorem.

Label the nodes, in descending order, $\nu_0, \nu_1, \ldots, \nu_{\log N}$ ($\nu_{\log N}$ is a leaf). Define $q := \log^{1/d} N$ and assume, for notational convenience, that $q \in \mathbb{N}$. For $z \in \{0, 1, \ldots, d-1\}$, define the collection of z-useful edges $E_z := \left\{(aq^z, \lceil \frac{a}{q} \rceil q^{z+1}) \ : \ q \nmid a, 0 \leq aq^z \leq \log N\right\}$, and let E_z^* be the smallest set containing E_z that satisfies the shrinkage property.

Notice that, for a fixed z, E_z has exactly one edge starting at each node. E_z^* has $\lceil \frac{a}{q} \rceil q^{z+1} - aq^z$ edges for node aq^z, so (defining, for $0 \leq x \leq \log N$, $\overline{x} \in \mathbb{Z}_q$ so $i \equiv \overline{x} \pmod q$) E_z^* has

$$\sum_{a=0}^{q^{d-z}} q^z(q - \overline{a}) - \sum_{x=0}^{q^{d-z-1}} q^z(q - \overline{xq}) = \frac{1}{2}q^d(q-1) < \frac{1}{2}q^{d+1} = \frac{1}{2}\log^{1+1/d} N$$

total edges. (The first sum does not take into account that $q \nmid a$, hence the subtraction of the second sum.) Our complete edge set will be the disjoint union $E = \cup_{z=0}^{d-1} E_z^*$. So $|E| = \sum_{z=0}^{d-1} |E_z^*| = \frac{1}{2} d \log^{1+1/d} N$, as required by the theorem. We now only need to see that this edge set contains a path of length at most d between any pair of nodes.

Let arbitrary $i < j$ be given and set $x \in \{0, 1, \ldots, d-1\}$ so that $q^x < j - i \leq q^{x+1}$. Set c_z to be the node labelled $\left\lceil \frac{i}{q^z} \right\rceil q^z$ (for $z \in \{0, 1, \ldots, x-1\}$). The sequence of labels is non-decreasing; if a repetition occurs, we will only include the last node of the repetition in our sequence. Now notice, for each $z \in \{1, 2, \ldots, x-1\}$, that there is an edge in E_z^* from c_{z-1} to c_z. Since $c_0 = \nu_i$ and since there is an edge from c_{x-1} to ν_j in E_{x-1}^*, we know that $(c_0 = \nu_i, c_1, \ldots, c_{x-1}, \nu_j)$ is a path from ν_i to ν_j (using edges in E) with at most $x + 1 \leq (d-1) + 1 = d$ edges. □

Halevy and Shamir make sure to point out that it requires an astronomical number of users in order to see the advantages of the general LSD scheme (over the basic LSD scheme).

We note that, for a fixed N, the number of keys per user $\frac{1}{2} d \log^{1+1/d} N$ is minimized at $d = \ln \log N$ (in Table 2, this is called scheme (j′)); the number of keys, then, equals $\frac{e}{2}(\log N)(\ln \log N)$ (where e is the base of the natural logarithm).

Alon [1] has shown that this graph-theoretic model has a lower bound on the number of required edges of $\Omega\left(\frac{1}{d^2} \log^{1+1/d} N\right)$, proving that the general LSD scheme is close to optimal (for schemes that use this graph-theoretic model).

Mironov [5] analyzes the graph problem more precisely, using a recurrence relation to construct and algorithm to find exact values for the optimal number of keys (for a given d and $\log N$). He includes a table with exact values for $d \leq 10$ and even-valued $\log N \leq 50$; the table naturally lead to the conjecture that each user (if the graph-theoretic model is used) must store at least $\Omega\left(\log^{1+1/d} N\right)$ keys.

Halevy and Shamir also mention a model called *IE trees* (inclusion-exclusion trees), where membership in $\overline{\mathcal{R}}$ is defined by labelling each vertex with "+", "−", or nothing (the root is always unlabelled), so a leaf is included in $\overline{\mathcal{R}}$ if and only if the nearest signed (*i.e.*, one with a + or a −) ancestor is a +. This has several natural applications, most notably situations arising from legal documents containing clauses with exceptions to exceptions (and more).

Their IE scheme is a modification of the standard subtree-difference (or LSD) scheme only by changing the Cover algorithm; the number of keys will be the same as whichever variant (scheme (h), (i), or (j)) we are changing the Cover algorithm of.

Cover continues the following until there are no nodes labelled with a +: find the +-marked node ν farthest from the root. If there are no nodes marked with a − below ν, add $S_{i,j}$ to the

cover, where ν_i is ν's parent and ν_j is ν's sibling; otherwise, call COVER on the subtree rooted at ν. In either case, remove the labels on all nodes in the subtree rooted by ν.

At every step, if the + at the then-root has no descendants marked −, we add 1 node to the cover; otherwise, the number of subsets we add is at most twice the number of − nodes below the +. As all the signs are then removed, we note that $|B| \leq 2M + P$, where there are M nodes labelled − and P nodes labelled +. If one of the LSD schemes is used, then some of the $S_{i,j}$ may not be available, and the usual multiplier applies: $|B| \leq d(2M + P)$.

Lastly, we note that the security of all of the LSD variants is at least as strong as that of the original subtree-difference scheme, as any adversarial coalition of users will have a subcollection of the keys they would have in the original scheme.

4.4 Traitor Tracing

Here we present a cursory discussion of traitor tracing, as it appears in [6].

Naor, Naor, and Lotspiech [6] present a tracing algorithm that works with a certain class of subset-cover scheme; to describe that class, we need a definition:

4.4.1 Bifurcation Value

Definition 4.5. A subset-cover broadcast-encryption scheme is said to have a *bifurcation value* of β if, for any subsets S_i in the subset collection $\{S_i\}_{i=1}^{w}$, it is possible to find two sets, S_a and S_b, that partition S_i such that $|S_a| \leq |S_b| \leq |S_i|/\beta$.

We will describe a tracing scheme that works for any subset-cover scheme with a constant (*i.e.*, independent of N) bifurcation value. Of course, its run time will depend on β. We now take a moment to analyze the bifurcation value of the practical subtree-difference schemes we have seen, schemes (g), (h), and (i).

In the complete-subtree scheme, any S_i can be split evenly in half by taking S_a and S_b as the two subtrees rooted at S_i's children. Hence, scheme (g) has a bifurcation value $\beta = 2$, the maximal (and, we will see later, the optimal) β. Let $S_{i,j}$ in the subtree-difference scheme be given. If ν_j is a child of ν_i, we have a complete subtree, so $S_{i,j}$ can be split as in the complete-subtree scheme. Otherwise, let ν_k denote the ancestor of ν_j that is a child of ν_i. Notice that $S_{i,k}$ and $S_{k,j}$ form a partition. In general, $|S_{a,b}| = \left(2^{D(\nu_b)-D(\nu_a)} - 1\right) 2^{D(\nu_a)+1}$. In the worst case (when ν_k is ν_j's parent), $|S_{i,k}| = 2|S_{k,j}| = (2/3)|S_{i,j}|$. Thus, for the subtree-difference scheme, $\beta = 3/2$.

In the basic LSD scheme, for a (useful) set $S_{i,j}$, if $qa < D(\nu_i) < D(\nu_j) \leq q(a+1)$ for some a, then we can use the same technique as with the subtree-difference method (if ν_j is a child of ν_i, we can split as in the complete-subtree method; if not, then $S_{i,k}$ and $S_{k,j}$ are useful). Otherwise, $q|D(\nu_i)$ and $D(\nu_j) - D(\nu_i) > q$. Choose k so ν_k is a child of ν_i and an ancestor of j and $D(\nu_k) = D(\nu_i) + q$. Notice that $S_{i,k}$ and $S_{k,j}$ are useful. In either case, $|S_{i,k}|/|S_{i,j}|$ is no worse than 2/3 (a bound that is achieved, again, when ν_k is ν_j's parent), so, for the basic LSD scheme, $\beta = 3/2$.

4.4.2 Subset-Tracing Algorithm

Before constructing the otherwise-simple traitor-tracing algorithm, we discuss subset-tracing algorithms. The subset-tracing algorithm found in [6] forms the core of the traitor-tracing algorithm we will build in Section 4.4.3.

Definition 4.6. For $p \in [0, 1]$, we say that a decoder box is *p-functional* (alternatively, the box has a *functionality of p*) *on a cover* $\{S_{i_j}\}_{j=1}^{m}$ if the box correctly decodes messages with probability at

least p. The probability is taken over the randomness of the box and a uniformly chosen random message.

Notice that if, for every possible $\mathcal{R}$, a box is p-functional on the cover induced by $\mathcal{R}$, then the box is p-functional (as defined in Section 2.2).

Definition 4.7. An algorithm $\mathcal{A}$ is called an *q-subset-tracing algorithm* if, when given a cover $\{S_{i_j}\}_{j=1}^{m}$ and a decoding box, $\mathcal{A}$ outputs $j \in \{1, 2, \ldots, m\}$ such that S_{i_j} contains a traitor or $\mathcal{A}$ outputs DISABLED and the box has functionality less than q on this particular cover.

The following theorem describes the q-subset-tracing algorithm that can be found in [6]:

Theorem 4.8. *For $q > \frac{1}{2}$, there exists a q-subset-tracing algorithm that is successful at least $1 - \varepsilon$ of the time and runs in time $O\big(m^2 \left(\log^3 m\right) \left(\log \frac{1}{\varepsilon}\right)\big)$.*

Their algorithm relies on finding a $j \in \{1, 2, \ldots, m\}$ for which,

Define p_j (for $j \in \{0, 1, \ldots, m\}$ to be the probability that the box outputs a successful decryption when the first j header parts $E_{L_{i_1}}(K), E_{L_{i_2}}(K), \ldots, E_{L_{i_j}}(K)$ are replaced with encryptions of random values. Notice that, if the illegal box is q-functional, then $p_0 = q$ and $p_m = \frac{1}{|\mathcal{K}|} \approx 0$. Since this means that one of the $|p_j - p_{j-1}|$ must be at least q/m, the algorithm seeks that through appropriate sampling with enough samples. The key to the result is that, then the algorithm reports knows that either the box has broken E, the box has broken the key-assignment method, or a traitor lies in S_{i_j}.

4.4.3 Traitor-Tracing Algorithm

Now we show how to turn any q-subset-tracing algorithm (*e.g.*, the one found in [6]) into a traitor-tracing algorithm:

Theorem 4.9. *Let $\mathcal{A}$ be a q-subset-tracing algorithm that works on covers of size up to $m + t \log_\beta N$ at least $1 - \varepsilon$ of the time. If $\mathcal{A}$ runs in time $\tau = \tau(m, \varepsilon, q)$, then we can construct a traitor-tracing algorithm for a subset-cover scheme with bifurcation value β that discovers or disables t traitors at least $1 - \varepsilon t \log_\beta N$ of the time and runs in time $O(\tau t \log_\beta N)$.*

Proof. Here we explicitly construct such a traitor-tracing algorithm:

Initialize $\mathcal{S} := \{S_{i_j}\}_{j=1}^{m}$. While $\mathcal{A}(\mathcal{S})$ outputs an index j, either output the unique element of S_{i_j} (if there is one) or replace S_{i_j} in $\mathcal{S}$ with two sets that form a bifurcation of S_{i_j} and run $\mathcal{A}$ again.

If we stop the above loop and have not output an element of $\mathcal{U}$ (from a singleton S_{i_j}), $\mathcal{A}(\mathcal{S})$ must have output DISABLED, so we can output $\mathcal{A}(\mathcal{S})$.

Notice that in the above algorithm, since the S_{i_j} are disjoint, a traitor can only force $\log_\beta N$ bifurcations until $\mathcal{A}$ outputs a singleton (or DISABLED). Hence, if there are t traitors, the traitor-tracing algorithm described here will need to run the q-subset-tracing algorithm $O(t \log_\beta N)$ times, each time on covers of size no larger than $m + t \log_\beta N$. □

References

[1] Noga Alon, "Monotone Broadcast Digraphs (comment; draft)", unpublished manuscript, 2002.

[2] Shimshon Berkovits, "How to Broadcast a Secret", *Advances in Cryptology: EUROCRYPT '91 (LNCS 547)*, pp. 535–541, 1991.

[3] Amos Fiat and Moni Naor, "Broadcast Encryption", *Advances in Cryptology: CRYPTO 1993 (LNCS 773)*, pp. 480–491, 1993. (A newer (extended) version of this paper can be found at: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/broad_abs.html`)

[4] Dani Halevy and Adi Shamir, "The LSD Broadcast Encryption Scheme", *Advances in Cryptology: CRYPTO 2002 (LNCS 2442)*, pp. 47–60, 2002.

[5] Ilya Mironov, "On Exact Solution to a Combinatorial Problem", unpublished manuscript, 2002.

[6] Dalit Naor, Moni Naor, and Jeff Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", *Advances in Cryptology: CRYPTO 2001 (LNCS 2139)*, pp. 41–62, 2001. (Newer version: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/2nl.html`)