Key Management for Restricted Multicast Using Broadcast Encryption

Michel Abdalla[1] and Yuval Shavitt[2] and Avishai Wool[3]

***Abstract*— The problem we address is how to communicate securely with a set of users (the target set) over an insecure broadcast channel. This problem occurs in two application domains: satellite/cable pay TV, and the Internet MBone. In these systems, the parameters of major concern are the number of key transmissions, and the number of keys held by each receiver. In the Internet domain, previous schemes suggest building a separate key tree for each multicast program, thus incurring a setup cost of at least $k \log k$ per program for target sets of size k. In the pay TV domain, a single key structure is used for all programs, but known theoretical bounds show that either very long transmissions are required, or that each receiver needs to keep prohibitively many keys.**

Our approach is targeted at both domains. Our schemes maintain a single key structure that requires each receiver to keep only a logarithmic number of establishment keys for its entire lifetime. At the same time our schemes admit low numbers of transmissions. In order to achieve these goals, and to break away from the theoretical bounds, we allow a controlled number of users outside the target set to occasionally receive the multicast. This relaxation is appropriate for many scenarios in which the encryption is used to force consumers to pay for a service, rather than to withhold sensitive information. For this purpose, we introduce *f-redundant* establishment key allocations, which guarantee that the total number of recipients is no more than f times the number of intended recipients. We measure the performance of such schemes by the number of key transmissions they require, by their redundancy f, and by the probability that a user outside the target set (a free-rider) will be able to decrypt the multicast. We prove a new lower bound, present several new establishment key allocations, and evaluate our schemes' performance by extensive simulation.

1 Introduction

The domain we consider in this paper is that of broadcast applications where the transmissions need to be encrypted. The examples we consider are a broadband digital TV network [18], broadcasting either via satellite or via cable, and Internet secure multicast [25], e.g., via the MBone [6].

In the context of **pay TV**, the *head-end* occasionally needs to multicast an encrypted message to some subset of users (called the target set) using the broadcast channel. Each network user has a *set-top terminal* (STT) which receives the encrypted broadcast and decrypts the message, if the user is entitled to it. For this purpose the STT securely stores the user's secret keys, which we refer to as establishment keys. Because of extensive piracy [19], the STTs need to contain a secure chip which includes secure memory for key storage. This memory should be non-volatile, and tamper-resistant, so the pirates will find it difficult to read its contents. As a result of these requirements, STTs have severely limited secure memory, typically in the range of a few Kilobytes.

Earlier work on broadcast encryption (cf. [8]) was motivated by the need to transmit the key for the next billing period or the key for the next pay-per-view event, in-band with the broadcast, since STTs only had uni-directional communications capabilities. The implicit assumption was that users sign up for various services using a separate channel, such as by calling the service provider over the phone. In such applications it is reasonable to assume that the target set is almost all the population, and there are only small number of excluded users. Moreover, it is crucial that users outside the target set are not able to decrypt the message since it has a high monetary value, e.g., the cost of a month's subscription.

However, current STTs typically follow designs such as [4] which allow bi-directional communication, where the uplink uses an internal modem and a phone line, or a cable modem. These new STTs upload the users' requests and download next month's keys via a callback mechanism, and not through the broadcast channel. This technological trend would seem to invalidate the necessity for broadcast encryption schemes completely. We argue that this is not the case—there are other applications where broadcast encryption is necessary, such as multicasting electronic coupons, promotional material, and low-cost pay-per-view events. Such applications need to multicast short-lived, low value messages that are not worth the overhead of communicating with each user individually. In such applications, though, the requirements from the solution are slightly different. On the one hand, it is no longer crucial that only users in the target set receive the message, as long as the number of free-riders is controlled. On the other hand, it is no longer reasonable to assume anything about the size of the target set.

Multicast in the **Internet** is a service that is bound to become more and more popular. Audio and video are two most talked about applications, but there are diverse data applications that can benefit from multicast, such as news updates, stock quotes, etc. Some of the multicast appli-

A preliminary version of this paper was presented at Financial Cryptography'99, Feb. 1999.

[1]Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, mabdalla@cs.ucsd.edu. Supported in part by CAPES under Grant BEX3019/95-2. Work done while author was visiting Bell Laboratories.

[2]Bell Laboratories, Lucent Technologies, 101 Crawfords Corner Rd., Holmdel, NJ 07733, shavitt@ieee.org.

[3]Bell Laboratories, Lucent Technologies, 700 Mountain Ave., Murray Hill, NJ 07974, yash@acm.org.

cations will be broadcasted freely, while others will be for pay. Once such pay-multicast services are deployed on the Internet, they would face similar issues faced by the pay-TV industry: High-value transmissions would need to be encrypted, and only paying customers should have the decryption keys.

In the past years, several suggestions for Internet multicast key-management architectures were proposed [21, 26, 25]. These proposals do not specify whether the decryption keys are to be held in a tamper-resistant hardware module or in the receiving hosts' insecure memory. We speculate that if Internet multicasts are to carry information with significant monetary value, then the service providers will encounter piracy. And in the Internet, the service providers lose the main advantage they have over the pirates in the pay-TV industry: The pirates have access to a high bandwidth, cheap, and world-wide, distribution channel—the same Internet that the service providers use (see [14] for a discussion of these issues). For this reason, we argue that successful Internet pay-multicast services would probably require users to keep keys in a secure STT-like device. But regardless of how Internet multicast keys are to be stored, it is certainly desirable to keep their number low.

For the rest of this paper, we use the term *"receiver"*, for both pay-TV STTs and Internet-multicast key stores (implemented either in a secure module or in the host memory).

1.1 Related Work

Fiat and Naor [8] were first to introduce broadcast encryption (in the context of pay-TV). They suggested methods of securely broadcasting key information such that only a selected set of users can decrypt this information while coalitions of up to k other users can learn nothing, either in the information-theoretic sense, or under a computational security model. Their schemes, though, required impractical numbers of keys to be stored in the receivers. Extensions to this basic work can be found in [1, 2, 24].

Recently Luby and Staddon [17] studied the trade-off between the transmission length and the number of keys stored in the receivers. They assumed a security model in which encryptions cannot be broken, i.e., only users that have a correct key can decrypt the message. We adopt the same security model. Their work still addressed fixed-size target sets, which are assumed to be either very large or very small, and no user outside the target set is allowed to be able to decrypt the message. A main part of their work is a disillusioning lower bound, showing that either the transmission will be very long or a prohibitive number of keys needs to be stored in the receivers.

A related line of work goes under the title of "tracing traitors" [3, 22]. The goal is to identify some of the users that leak their keys, once a cloned receiver is found. This is achieved by controlling which keys are stored in each receiver, in a way that the combination of keys in a cloned receiver would necessarily point to at least one traitor.

Key management schemes for encrypted broadcast networks, which give the vendor the flexibility to offer program packages of various sizes to the users, can be found in [27]. The problem of tracking the location of receivers in order to prevent customers from moving a receiver from, e.g., a home to a bar, is addressed in [9].

The Iolus project [21] was the first serious attempt to propose a framework for secure Internet multicast. The Iolus framework is based on a hierarchical tree structure. At higher layers, group servers communicate with each other using secure multicast. At lower layers, secure communication is exchanged between group members. Special servers are needed to handle the join/leave operation in each level.

Wong et al. [26] were the first to suggest a *key management* scheme for secure multicast in the Internet. Their work influenced the network working group of the IETF in the form of a recent RFC [25]. Their solution is based on a hierarchy of keys that is built with the group of currently paying customers. The customers of each service (a movie channel, a stock quote service, a news bulletin) are the leaves of a balanced tree, where each node of the tree corresponds to a group key; the group members are the descendents of that node. This solution achieves a logarithmic cost for join/leave operations. However, the tree is built per program (or per group), and as such incurs a high overhead when many different programs are multicast.

1.2 Contributions

Our starting point is the observation that the requirement "no users outside the target set can decrypt the message" is too strict for many applications. For instance, for the purposes of multicasting electronic coupons, it may be enough to guarantee that the recipient set contains the target set, and that the total number of recipients is no more than f times the size of the target set. Service providers can afford a small potential increase in the number of redeemed coupons, as long as this simplifies their operations and lowers their cost. We call establishment key allocation schemes that provide such guarantees "f-redundant broadcast encryption schemes". Relaxing the requirements in this way allows us to depart from the lower bounds of [17].

On the other hand, we have a more ambitious goal when it comes to possible target sets. Unlike earlier work, we require our schemes to be able to multicast to *any* target set, not just those target sets of very small or very large cardinality.

We concentrate on schemes which store only a small number of keys in each receiver. For systems with several million users, it is reasonable to require the maximum number of keys per user to be $O(\log n)$, where n is the total number of users, or at most $O(n^\epsilon)$, where, say, $\epsilon \leq 1/4$.

Subject to these constraints, we are interested in several measures of the quality of an establishment key allocation. The first is the number of transmissions, t: we can always attain our requirements trivially if we assign each receiver a unique key, but then we suffer a very high number of transmissions. The second parameter, which we call *opportunity*, is the proportion of free-riders in the population outside the target set. The opportunity measures the incentive a customer has to avoid paying (in cheap pay-per-view type services). If the opportunity is very high, close to 1, there is no incentive for customers to pay, as they can almost surely get a free ride.

After discussing the basic trade-offs associated with the problem, we present some simple examples, that show the problem difficulty. We then prove a new lower bound on the tradeoff between the transmission length and the number of keys stored per receiver, a lower bound that incorporates the f-redundancy of our establishment key allocations. We show that the f-redundancy gives us a substantial gain: For the same number of transmissions t we can hope for only $\exp(\Omega(n/tf))$ keys per receiver, whereas the bound of [17] is $\exp(\Omega(n/t))$.

We then present several establishment key allocation constructions, and an approximation algorithm that finds a key cover with minimal number of transmissions, for any given target set of users. Since this problem is similar to the minimum set cover problem, that is known to be NP-hard, we cannot expect to find an optimal solution efficiently. Instead, we use a greedy approximation algorithm to find good key covers. We conducted an extensive simulation study of the problem, from which we present only the interesting results.

Finally, we discuss the practical aspects of using our scheme for key management of secure multicast on the Internet. We propose a single key management structure for all the services in a given infrastructure (e.g., one key structure for the entire MBone [6]). The cost of building this key management structure is logarithmic in the size of the total user population, but it is now usable for all the services provided on this infrastructure, making it much cheaper than the "separate tree per group" proposed by [26, 25]. We discuss how to build and maintain our structure incrementally, as users are added or dropped from the MBone. We also discuss the practical issues of how to manage the key transmissions in a dynamic environment, where paying users join and leave a specific program while it is in progress. We show that such a dynamic environment further encourages users to pay.

Organization: In the next section we formally define the problem and the various parameters we are interested in. In Section 3 we show some simple solutions. In Section 4 we prove our new lower bound on the trade-off between the number of keys per user, the redundancy factor, and the transmission length. In Section 5 we discuss how to find which keys to use given an establishment key allocation. In Section 6 we show our schemes and the results of their performance evaluation. In Section 7 we discuss how to adapt our schemes to dynamic Internet environments, and evaluate their performance in such environments. We conclude in Section 8.

2 Definitions and Model

Let $\mathcal{U}$ be the set of all the receivers (i.e., receivers connected to a head-end), with $|\mathcal{U}| = n$. We use K to denote the *target set*, i.e., the set of paying customers, and denote its size by $|K| = k$.

We describe the allocation of the establishment keys by a collection $\mathcal{S} = \{S_1, S_2, \ldots\}$ of *key sets* such that $\cup S_i = \mathcal{U}$. We associate a unique establishment key e_i with each set $S_i \in \mathcal{S}$. A key e_i is stored in the secure memory of every receiver $u \in S_i$. Hence the number of keys a receiver $u \in \mathcal{U}$ stores is equal to the number of sets $S_i \in \mathcal{S}$ it belongs to. Formally,

Definition 2.1 *Let $\mathcal{S}$ be an establishment key allocation. The* degree *of a receiver u is $deg(u) = |\{i : S_i \ni u\}|$. The degree of a collection $\mathcal{S}$ is $deg(\mathcal{S}) = \max_{u \in \mathcal{U}} deg(u)$.*

Definition 2.2 *Given a target set K, a* key cover *of K is a collection of sets $S_i \in \mathcal{S}$ whose union contains K:*

$$\mathcal{C}(K) \subseteq \mathcal{S} \text{ such that } K \subseteq \cup_{S_i \in \mathcal{C}(K)} S_i.$$

The minimal key cover *is $\mathcal{C}_{\min}(K) = \mathcal{C}(K)$ for which $|\mathcal{C}(K)|$ is minimal.*

Suppose the head-end needs to send a message μ to all the members of a target set K. Given any key cover $\mathcal{C}(K)$, the head end encrypts μ using the establishment keys e_i corresponding to the sets $S_i \in \mathcal{C}(K)$, and broadcasts each encryption separately.[4].

Definition 2.3 *We denote the best possible number of transmissions that the head-end can use for a target set K by $t_K = |\mathcal{C}_{\min}(K)|$. Thus the worst case number of transmissions is $t_{\max}(\mathcal{S}) = \max_K t_K$.*

In order to define the redundancy and opportunity measures we need the following technical definition.

Definition 2.4 *We denote the set of recipients of a given key cover $\mathcal{C}(K)$ by $R_{\mathcal{C}}(K) = \cup\{S_i \in \mathcal{C}(K)\}$ and the total number of recipients by $r_{\mathcal{C}}(K) = |R_{\mathcal{C}}(K)|$.*

By the definition of a key cover $\mathcal{C}(K)$, every member of the target set K has at least one of the keys used to encrypt μ. However, other receivers outside K usually exist, which are also capable of decrypting the message. All our establishment key allocations are constructed with a worst case guarantee that there are never too many of these free-riders. Formally,

Definition 2.5 *An establishment key allocation $\mathcal{S}$ is said to be* f-redundant *if*

$$\frac{r_{\mathcal{C}}(K)}{k} \leq f$$

for every $K \subseteq \mathcal{U}$ with $|K| = k$.

A variant measure of redundancy is the *actual redundancy f_a*, which is the ratio between the non-paying and paying recipients. We are interested in the average case f_a, so we define it as a function of the target set K. Formally,

[4] This method was called the OR protocol in [17].

$\mathcal{S}$	$deg(\mathcal{S})$	$t_{\max}(\mathcal{S})$	f	η
$\{\mathcal{U}\}$	**1**	**1**	n	1
$\{\{1\},\ldots,\{n\}\}$	**1**	n	**1**	**0**
$2^{\mathcal{U}}$	2^{n-1}	**1**	**1**	**0**

Table 1: A summary of some simple examples. Bold numerals indicate an optimal parameter.

Definition 2.6 *For a target set K with $|K| = k$ the* actual redundancy *is $f_a = \frac{r_{\mathcal{C}}(K)-k}{k}$.*

If $\mathcal{S}$ guarantees a worst case redundancy factor f, then $0 \le f_a \le f - 1$ for any target set K.

Finally, we define the opportunity, η, as the proportion of non-paying recipients (free-riders) in the non-paying population ($0 \le \eta \le 1$). The opportunity measures the incentive a customer has to avoid paying (e.g., in cheap pay-per-view type services). Again, this is a function of the target set K.

Definition 2.7 *For a target set K with $|K| = k$ the* opportunity *is $\eta = \frac{r_{\mathcal{C}}(K)-k}{n-k}$.*

3 Simple Examples

To demonstrate our definitions and the trade-offs associated with the problem let us examine some simple solutions for the problem (which are similar to those in [26]). See Table 1 for a summary of the examples.

Example 3.1 *The "always broadcast" solution: $\mathcal{S} = \{\mathcal{U}\}$.*

Both the degree, $deg(\mathcal{S})$, and the number of transmissions, $t_{\max}(\mathcal{S})$, required to distribute the message are optimal and equal to 1 in this case. However, the redundancy is $f = n$ in the worst case and the opportunity, η, is always 1. The last two parameters are very bad since the system gives no incentive for a customer to pay for a program; a single paying customer enables the entire population a free ride.

Example 3.2 *The "key per user" solution: $\mathcal{S} = \{\{1\},\{2\},\ldots,\{n\}\}$.*

Here the degree $deg(\mathcal{S}) = 1$ is optimal, and so are the redundancy $f = 1$, and the opportunity $\eta = 0$. However, the number of transmissions is a very poor $t_{\max}(\mathcal{S}) = n$.

Example 3.3 *The "all possible sets" solution: $\mathcal{S} = 2^{\mathcal{U}}$.*

The degree here is an impractical $deg(\mathcal{S}) = 2^{n-1}$, however, all the other parameters are optimal: $t_{\max}(\mathcal{S}) = 1$, $f = 1$, and $\eta = 0$. This is because every possible target set K has its own designated key.

4 The Lower Bound

4.1 Tools

Before presenting our lower bound on the degree of an f-redundant establishment key allocation, we need to introduce some definitions and results which we use in the proof.

We start with *covering designs*, which are a class of combinatorial block designs. A succinct description of covering designs can be found in [5, Ch. IV.8]. A more detailed survey is [20].

Definition 4.1 *A k-(n,d)* covering design *is a collection of d-sets* (blocks) *$\mathcal{D} = \{D_1,\ldots,D_\ell\}$ over a universe of n elements, such that every k-set of elements is contained in at least one block.*

Definition 4.2 *The* covering number *$C(n,d,k)$ is the minimum number of blocks in any k-(n,d) covering design.*

Theorem 4.3 (Schönheim bound) [23] *$C(n,d,k) \ge L(n,d,k)$, where*

$$L(n,d,k) = \left\lceil \frac{n}{d} \left\lceil \frac{n-1}{d-1} \cdots \left\lceil \frac{n-k+1}{d-k+1} \right\rceil \right\rceil \right\rceil \ge \binom{n}{k} \Big/ \binom{d}{k}.$$

We also rely on the following result of Luby and Staddon, which addresses *strict* broadcast encryption protocols.

Definition 4.4 *An establishment key allocation $\mathcal{S}$ is called* strict *for a collection of target sets $\mathcal{D}$ if the sets in $\mathcal{D}$ can be covered without redundancy. Formally, $R_{\mathcal{C}}(D) = D$ for all $D \in \mathcal{D}$.*

Theorem 4.5 [17] *Let $\mathcal{D} = \{D_1,\ldots,D_\ell\}$ be a collection of target sets, with $|D_i| \ge d$ for all $D_i \in \mathcal{D}$. Then any establishment key allocation $\mathcal{S}$ which is strict for $\mathcal{D}$, and which can transmit to any $D_i \in \mathcal{D}$ using at most t transmissions, must have*

$$deg(\mathcal{S}) \ge \left(\frac{\ell^{1/t}}{t} - 1\right) \Big/ (n-d).$$

Remark: The precise statement we use here is a generalization of [17, Theorem 12]. In their original formulation the target sets D_i all have a cardinality of exactly d, and the collection $\mathcal{D}$ consists of all $\binom{n}{d}$ possible d-sets. However, their proof can be easily extended to any arbitrary collection of target sets, of cardinality d or larger.

4.2 The Bound

Theorem 4.6 *Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n, for which $t_{\max}(\mathcal{S}) = t$. Then*

$$deg(\mathcal{S}) \ge \max_{1 \le k \le n/f} \left(\frac{1}{t} \left[\binom{n}{k} \Big/ \binom{kf}{k} \right]^{1/t} - 1 \right) \Big/ (n-k).$$

Proof: For a target set K of size $|K| = k$, let $R(K)$ be the minimal possible recipient set for K (or one such set if many minimal recipient sets exist). Consider the collection of minimal recipient sets

$$\mathcal{D} = \{R(K) : |K| = k\}.$$

Note that covering K f-redundantly, using the $t' \le t$ key sets that define $R(K)$, is precisely equivalent to covering $R(K)$ *strictly* with (the same) t' key sets. Therefore we see that $\mathcal{S}$ is an establishment key allocation which is strict for $\mathcal{D}$, and can transmit to any $R(K) \in \mathcal{D}$ using at most t transmissions. Note also that, trivially, $|R(K)| \geq k$ for any $|K| = k$. Thus we can apply Theorem 4.5 to obtain

$$deg(\mathcal{S}) \geq \left(\frac{|\mathcal{D}|^{1/t}}{t} - 1 \right) \Big/ (n-k). \qquad (1)$$

By definition $|R(K)| \leq kf$ for all K, however, some sets $R(K) \in \mathcal{D}$ may have fewer than kf elements. Define a modified collection $\mathcal{D}'$ in which each $R(K) \in \mathcal{D}$ is replaced by some superset $\hat{R}(K) \supseteq R(K)$ with $|\hat{R}| = kf$. Note that $|\mathcal{D}'| \leq |\mathcal{D}|$ since $\hat{R}(K_1) = \hat{R}(K_2)$ is possible when $R(K_1) \neq R(K_2)$. But now $\mathcal{D}'$ is a k-(n, kf) covering design. Thus we can lower-bound its size by the Schönheim bound, Theorem 4.3, to obtain

$$|\mathcal{D}| \geq |\mathcal{D}'| \geq L(n, kf, k) \geq \binom{n}{k} \Big/ \binom{kf}{k}. \qquad (2)$$

Plugging (2) into (1) and maximizing the expression over the choice of k yields our result. ▮

Using standard estimations of binomial coefficients, and maximizing over k, we can obtain the following asymptotic estimate.

Corollary 4.7 *Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n, for which $t_{\max}(\mathcal{S}) = t$. Then $deg(\mathcal{S}) \geq \exp(\Omega(n/tf))$.* ▮

We therefore see that the f-redundancy gives us a substantial gain in the degree: the bound of [17] for strict establishment key allocations is $deg(\mathcal{S}) = \exp(\Omega(n/t))$. In other words, if we allow a redundancy factor of f we can hope to use only an f'th root of the number of keys required per receiver in a strict establishment key allocation for the same number of transmissions.

Theorem 4.6 and Corollary 4.7 give a lower bound on the required number of keys a receiver needs to store. As we said before, this is typically a small fixed value which we can reasonably model by $\log_2 n$ or n^ϵ. Thus we are more interested in the inverse lower bound, on the number of transmissions t. Asymptotically we can obtain the following bound.

Corollary 4.8 *Let $\mathcal{S}$ be an f-redundant establishment key allocation over a universe $\mathcal{U}$ of size n. Then*

$$t_{\max}(\mathcal{S}) \geq \begin{cases} \Omega\left(\frac{n}{f \log\log n}\right), & \textit{when } deg(\mathcal{S}) = O(\log n), \\ \Omega(\frac{n}{f \log n}), & \textit{when } deg(\mathcal{S}) = O(n^\epsilon). \end{cases}$$

The asymptotic bound of Corollary 4.8 hides the constants, and inverting Theorem 4.6 gives a rather unwieldy expression for the lower bound on t. Therefore, we choose to invert Theorem 4.6 numerically and to plot the result, as a function of the target set size k, in Figure 1. As we shall see in the sequel, the highest point on this curve ($t \approx 19$ for $n = 1024$) is significantly lower than our best constructions, which suffer from a worst case of $t_{\max}(\mathcal{S}) = 3n/8 = 384$ when $n = 1024$.

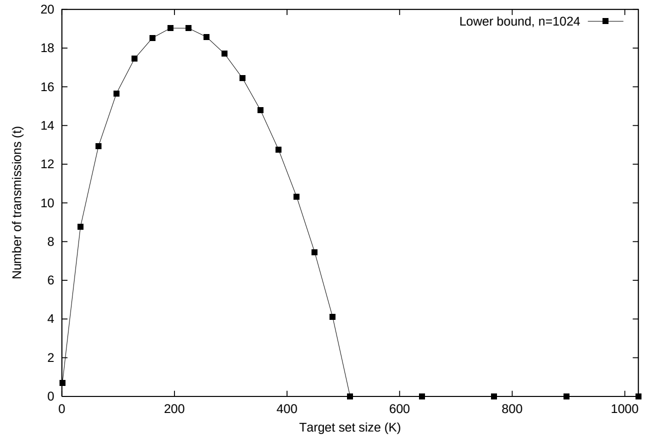

Figure 1: The lower bound for the number of transmissions (t) as a function of the target set size k, with $n = 1024$, $f = 2$, and $deg(\mathcal{S}) = \log_2 n$.

5 Finding a Good Key Cover

Input: Target set K,
establishment key allocation $\mathcal{S} = \{S_1, \ldots, S_m\}$.

---

0. $R \leftarrow \emptyset$; $\mathcal{C} \leftarrow \emptyset$
1. **Repeat**
2. $\quad \mathcal{A} \leftarrow \{S_i : \frac{|S_i \setminus R|}{|(K \cap S_i) \setminus R|} \leq f\}$.
3. $\quad A \leftarrow S_i \in \mathcal{A}$ which maximizes $|(K \cap S_i) \setminus R|$.
4. $\quad R \leftarrow R \cup A$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{A\}$.
5. **until** the candidate collection $\mathcal{A}$ is empty.
6. **return** R, $\mathcal{C}$.

Figure 2: Algorithm f-*Cover*

An f-redundant establishment key allocation guarantees that an f-redundant cover exists for every target set K. In particular, singleton target sets $K = \{u\}$ need to be addressed. Thus, $\mathcal{S}$ must include enough sets S_i with $|S_i| \leq f$ so that every user is contained in one of them. For simplicity, we shall assume that $\mathcal{S}$ contains the singletons themselves as sets, i.e., every receiver is assumed to hold one key that is unique to it.

Once we decide upon a particular f-redundant establishment key allocation $\mathcal{S}$, we still need to show an efficient algorithm to find an f-redundant key cover $\mathcal{C}(K)$ for every target set K. Among all possible f-redundant key covers that $\mathcal{S}$ allows, we would like to pick the best one. By "best" we mean here a cover that minimizes the number of transmissions t. Trying to minimize the actual redundancy f_a would lead to trivialities: Since we assumed that $\mathcal{S}$ contains all the singletons we can always achieve the optimal $f_a = 0$. Thus, for every target set K, we obtain the following optimization problem:
Input: A collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$ and a target set K.
Output: A sub-collection $\mathcal{C}_{\min}(K) \subseteq \mathcal{S}$ with minimal car-

dinality $|\mathcal{C}_{\min}(K)|$ such that $K \subseteq \cup\{S_i \in \mathcal{C}_{\min}(K)\}$ and $|\cup\{S_i \in \mathcal{C}_{\min}(K)\}|/|K| \leq f$.

This is a variation of the Set Cover problem [10], and thus an NP-hard optimization problem. We omit the formal reduction proving this. Moreover, it is known that no approximation algorithm exists for Set Cover with a worst case approximation ratio[5] better than $\ln n$ (unless NP has slightly super-polynomial time algorithms) [7].

On the positive side, the Set Cover problem admits a greedy algorithm, which achieves the best possible approximation ratio of $\ln n$ [13, 16]. Moreover, the greedy algorithm is extremely effective in practice, usually finding covers much closer to the optimum than its approximation ratio guarantees [11]. For this reason, our general algorithm *f-Cover* for choosing a key cover is an adaptation of the greedy algorithm. See Figure 2 for the details.

Theorem 5.1 *If $\{\{1\},\ldots,\{n\}\} \subseteq \mathcal{S}$ then algorithm f-Cover returns an f-redundant key cover of K for any target set K.*

Proof: The set R maintains the current cover in the algorithm. In every iteration, when a set S_i is added to the cover, $|S_i \setminus R|$ new users are covered, and $|(K \cap S_i) \setminus R|$ of them are target set members that were not included in the cover before. Note that we only add a set S_i if $\frac{|S_i \setminus R|}{|(K \cap S_i) \setminus R|} \leq f$ and that the sets $(S_i \setminus R)$ are disjoint for the S_i's chosen in different iterations. From these observations it is easy to prove that the f-redundancy is kept throughout the algorithm execution, and in particular, when the algorithm terminates. ∎

Remarks:

- The candidate set $\mathcal{A}$ needs to be re-calculated in each iteration, since a non-candidate set S_i may become a candidate (or vice versa) after some other S_j is added to the cover.
- It is easy to see that the time complexity of algorithm *f-Cover* is $O(m^2)$ where m is the number of sets in $\mathcal{S}$.
- In practice the computation time on a 433 MHz Intel Celeron processor was between 1ms and 17ms for N =1024, 2048, and 4096, and various values of k.

In order to make the algorithm even more efficient, we do not use it in its most general form. Instead, we split the establishment key allocation $\mathcal{S}$ into *levels*, each containing sets of the same size. Formally, we break $\mathcal{S}$ into $\mathcal{S} = \mathcal{S}^1 \cup \mathcal{S}^2 \cup \cdots$, such that $|S_i^\ell| = k_\ell$ for some k_ℓ and for all $S_i^\ell \in \mathcal{S}^\ell$. The algorithm is performed in phases, where only sets belonging to level $\mathcal{S}^\ell$ are considered in the candidate set $\mathcal{A}$ during in phase ℓ. The algorithm starts at the highest level, the one containing of the largest sets in $\mathcal{S}$. When $\mathcal{A}$ is empty at a certain level, the cover so far, R, and the covering sets, $\mathcal{C}$, are fed to the execution phase of the algorithm in the next (lower) level.

[5] [12] contains a good discussion of approximation algorithms and in particular a chapter on Set Cover.

6 Practical Solutions

6.1 Overview

Our basic goal is to construct an f-redundant establishment key allocation, namely to construct an $\mathcal{S}$ that will satisfy the following requirements: (i) the number of establishment keys per user (degree) is low; and (ii) $|R_{\mathcal{C}}(K)|/|K| \leq f$ for every target set $K \subseteq U$. Given such an establishment key allocation, we evaluate its performance with respect to the number of transmissions t, the actual redundancy f_a, and the opportunity η, using computer simulations.

We are interested in "average" performance, but since performance depends heavily on the target set size, k, we use it as a parameter in the simulations. Each data point for a target set size k in the graphs represents the mean of the relevant measure, averaged over r samples of k-sets chosen uniformly at random. We show the 95% confidence intervals[6] for each data point, unless the graphical height of the confidence intervals is very close to the size of the symbols depicted on the curves. We typically use $r = 25$ samples per data point.

Unless stated otherwise, we assume that the redundancy is $f = 2$. We also conducted experiments with other values of f but they showed qualitatively similar results.

6.2 The Tree Scheme

6.2.1 The Scheme's Description

A simple multi-level establishment key allocation is a balanced tree, that is built by recursively partitioning the sets of a high level into equally-sized, disjoint sets in the next level. Sets that form a partition of a single set, one level above them, are considered children of this set in the tree. The number of keys each receiver holds in this scheme is only $1 + \log_a n$, where a is the arity of the tree. In the sequel we always assume a binary tree ($a = 2$).

An important advantage of a tree scheme (besides its simplicity) is that the greedy algorithm of Figure 2 can easily be made to run in time linear in the size of the cover set, rather than in the total number of sets in the collection. The idea is to start at the root of the tree (the set $\mathcal{U}$) and then traverse it either in a DFS or in a BFS order. Whenever an f-redundant set is found, select it and ignore the subtree under it.

The problem with the tree scheme is its worst case behavior. Consider the case where $f = 2$ and the collection is a full binary tree. If the target set comprises $k = n/4$ users such that no two of them belong to a common set of size 4 or less, then we are forced to use $t = n/4$ transmissions. It is easy to see that this is the worst possible configuration.

The average behavior of the basic tree is substantially better than the worst case. Figure 3 shows the average number of transmissions on several variants of a tree for a population of $n = 1024$ users. We see from the "threshold

[6] A 95% confidence interval means that the population mean appears within the specified interval with probability 0.95. See [15] for a precise definition of confidence interval.

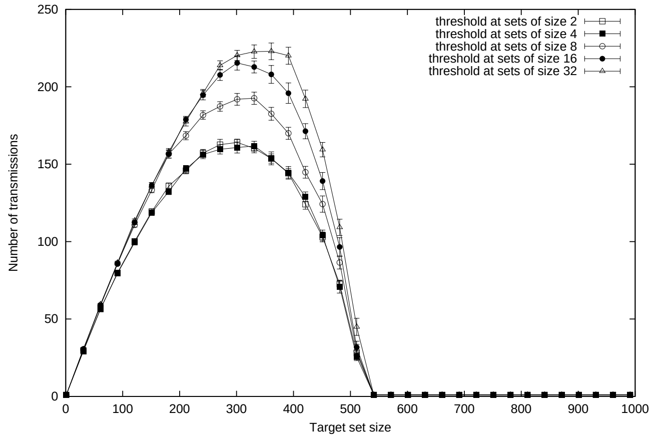

Figure 3: The effect of the "$\leq$" threshold T on the number of transmissions (t), for a tree with $n = 1024$.

at sets of size 2" curve in the figure that the peak of the average t is 164, which is 36% less than the worst case of 256. We explain this threshold and discuss the different variants of the tree in Section 6.2.2.

We conducted the same tests for larger populations and noticed that the qualitative behavior does not change significantly, thus we omit the details. Here we focus on simulations of small populations for another reason. We shall see in Section 6.4 that we can capitalize on the detailed understanding of small populations when we discuss partitioning large populations. Our results show that breaking a large population into small subgroups and solving the problem independently for each subgroup results in a good performance trade-off.

6.2.2 "$<$" or "$\leq$"?

A subtle issue in the execution of algorithm *f-Cover* is whether the inequality in step 2 is strict ($<$) or not ($\leq$). Assume that $f = 2$ and that the collection $\mathcal{S}$ is a full binary tree. If a set of size S_i with $|S_i| = 2$ is tested using non-strict inequality, and only one member of S_i is in the target set K, then S_i is selected as a candidate and may be part of the cover. However, using a strict inequality gives a better choice, which is to select the singleton containing that user, thereby reducing the actual redundancy without increasing the number of transmissions. On the other hand, using strict inequality for larger set sizes tends to increase the number of transmissions. So, intuitively, we would like to use "$<$" in the lowest levels of the tree, and use "$\leq$" for sets of size T or larger, for an appropriate threshold T. Figures 3, 4 and 5 compare the performance of a tree scheme when the threshold is varied. Note that the $T = 2$ curve, which we commented on before, represents using "$\leq$" everywhere.

The most striking graph is that of the actual redundancy (Figure 4). We see that when we use strict inequality in the level of the tree corresponding to sets of size 2 (i.e., the "$\leq$" threshold is $T = 4$) the actual redundancy, f_a, drops dramatically for target set sizes below $n/2$. At the same

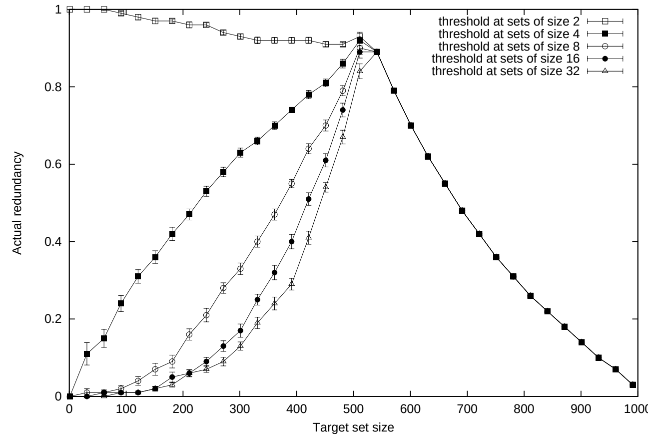

Figure 4: The effect of the "$\leq$" threshold T on the Actual redundancy (f_a) for a tree with $n = 1024$.

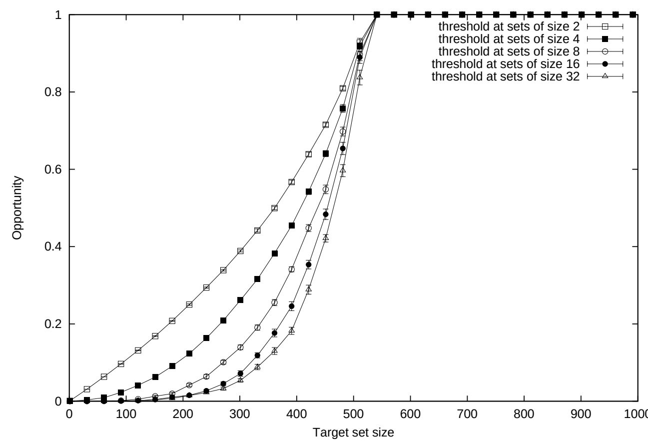

Figure 5: The effect of the "$\leq$" threshold T on the Opportunity (η), for a tree with $n = 1024$.

time, the number of transmissions, t, remains unchanged. There is also an improvement in the opportunity, η. Moving the threshold further up improves f_a and η at the cost of increasing t. We found out that, in most cases, and especially when extra keys are added (see below), it pays to set the threshold at $T = 8$ since the increase in t is very small while the gain in f_a and η is substantial. Thus, in all the following simulations we only use strict inequality for sets of size 4 and below.

Note that choosing $T = 8$ has an effect on the worst case performance since now $k = 3n/8$ users can be selected such that no four of them belong to a common set of size 8 and no three of them belong to a common set of size 4. As a result, we would be forced to use $t = 3n/8$ transmissions, all at the level corresponding to singleton sets.

When $T = 8$, the peak number of transmissions t is $193 \approx n/5$ (see Figure 3), which means a 50% improvement over the worst case performance of 384, and achieves actual redundancy that is always lower than 0.9. However, in most of the range the results are much better. In particular, if the interesting target set size range is below $k = n/5$, we

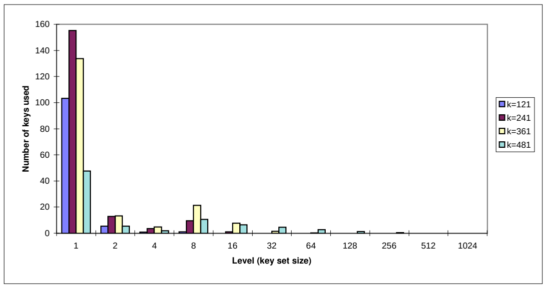

Figure 6: A histogram of the key sizes used for several target set sizes k, for $n = 1024$.

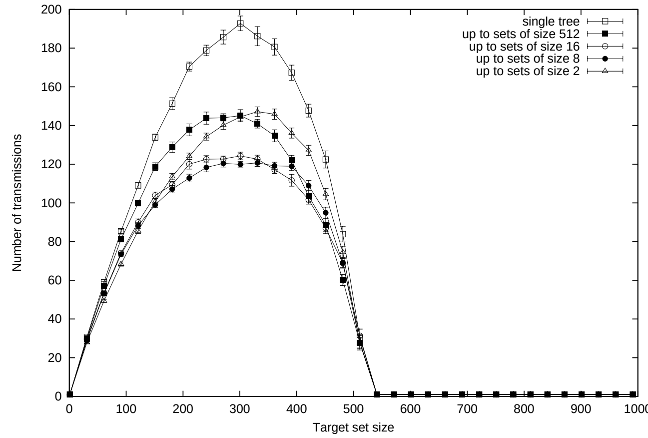

Figure 7: Number of transmissions (t) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

get $t < n/6$, $f_a < 0.16$, and $\eta < 0.04$.

6.3 Where Extra Keys are Effective

The basic tree scheme requires only $\log_2 n$ keys to be stored in each receiver. Therefore it is reasonable to consider schemes with slightly more keys: For populations of several millions, we can afford to keep twice or four times as many keys in a receiver.

In this section, we study schemes in which a tree is augmented by additional sets. The motivation for doing so is clear: by increasing the number of sets (and thereby keys), the probability of finding a smaller cover increases. We are interested in locating the levels where it best pays to add sets, subject to the constraints on the number of keys per receiver.

In order to generate the extra key sets, we start with a "level-degree" profile, which specifies how many keys each user should hold at each level. For a level with set size k, a degree of d implies that each user should belong to $d-1$ extra sets, in addition to the one basic tree set it belongs to at this level. Thus we need to be able to generate nd/k sets of size k, such that each user belongs to exactly d of them. We achieve this by randomly permuting the n users $d-1$ times, and for each random permutation we add the users in positions $(i-1)k+1, \ldots, ik$ as a set, for $i = 1, \ldots, n/k$.

A vivid explanation for the preferred placement of the extra keys can be found in the histogram in Figure 6. The histogram depicts the number of keys used from each level of sets, for target sets of four sizes. We used a population of $n = 1024$ users and a basic tree scheme with 11 levels. The histogram clearly shows that the small sets are the ones used most often. As the target set size grows, some larger key sets are also used. However, even when the target sets are $k = 241$ and $k = 361$, i.e., target sets requiring the highest number of transmissions, relatively few keys are used for sets of size 32 and up. Therefore it seems that adding key sets at the low levels of the tree is the right approach.

Figures 7, 8, and 9 depict the performance of an 11-level tree ($n = 1024$) augmented tree with 9 extra keys. This choice allows us to double the number of keys per level in all the intermediate levels ($1 < |S_i| < n$). Following the conclusions we draw from the key usage histogram in Figure 6, these extra keys are distributed as uniformly as possible among the levels from the bottom (couples) level up to some level ℓ. We varied ℓ in order to find the most effective distribution.

We first note that regardless of how the extra keys are distributed, the peak number of transmissions drops by at least 23% (from 193 down to 147 for the "up to sets of size 2" distribution) in comparison to a non-augmented tree.

Figure 7 shows that the best t is achieved by distributing the extra keys at the three lowest levels, i.e., adding couples, quadruplets, and octets. Adding sets of size 16 as well resulted in an almost identical performance. However, adding even larger sets gave significantly inferior performance. Figures 8 and 9 show that this improvement comes at the expense of an increase in f_a for small target set sizes, although the actual redundancy is still well below the guaranteed worst case of $f_a \leq f - 1$ ($= 1$ when $f = 2$).

In a similar experiment with 38 keys ($= 11 + 3 \times 9$) per user, the best t was achieved by spreading the keys among the lowest 4 levels (up to sets of size 16); the peak t for this experiment was about 94 transmissions, for target sets of size 271 ($= n/3.8$), which is 22% lower than the 121 achieved in Figure 7 by the "up to sets of size 8" distribution. We also ran the same experiments for larger and smaller values of n, with similar results. We omit the details.

Our conclusions from this set of experiments are that (a) adding a few extra keys per user substantially reduces the number of transmissions t, and (b) it pays to add these extra keys at the lower levels of the tree rather than to distribute them at higher levels as well.

6.4 Partitioning

The results in the previous sections suggest that keys are more "valuable" at the lower levels of the tree than at the higher levels. Thus, it seems reasonable to discard the keys of the largest sets (highest levels) altogether, and to use the additional key space for more lower level keys. We achieve

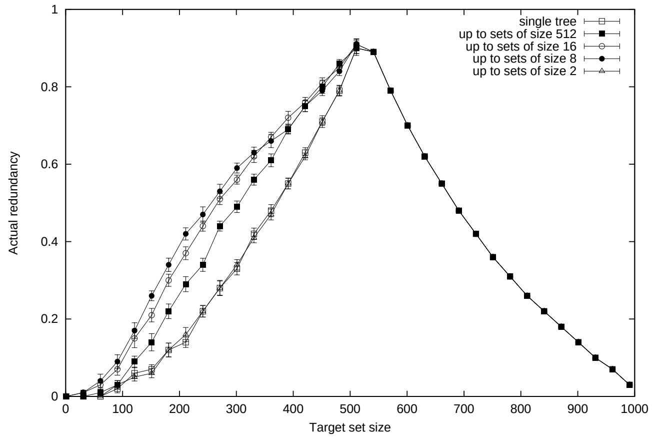

Figure 8: Actual redundancy (f_a) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

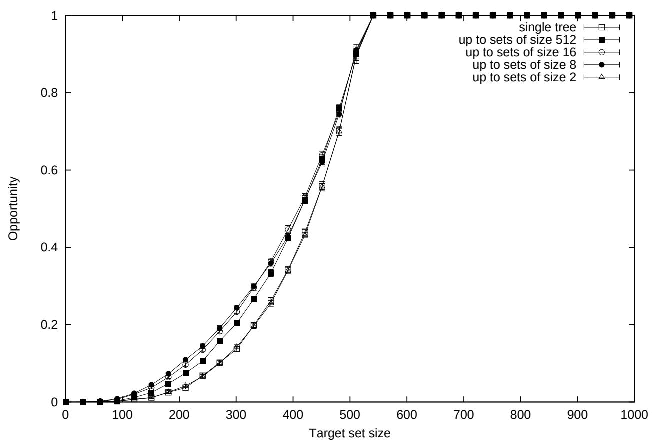

Figure 9: Opportunity (η) as a function of the target set size k, with $n = 1024$, $f = 2$, 11 levels, and 9 extra keys.

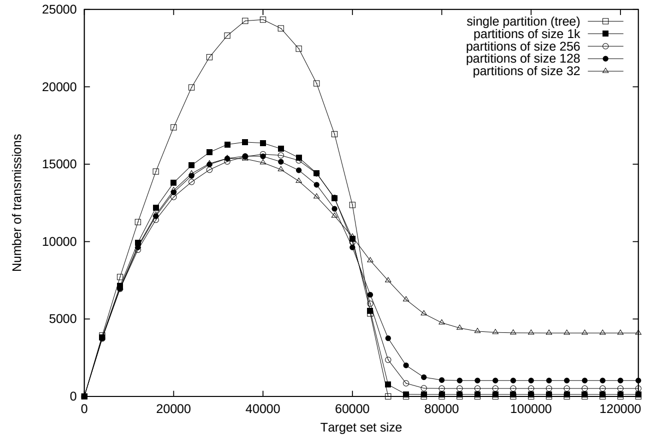

Figure 10: Number of transmissions (t) as a function of the target set size k, with $n = 128K$, $f = 2$, and 18 keys in total.

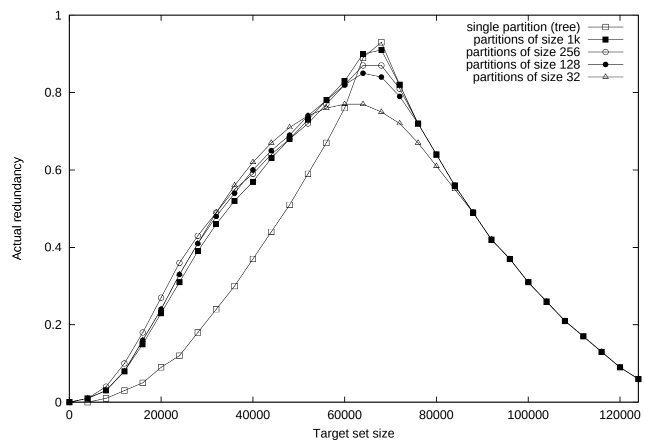

Figure 11: Actual redundancy (f_a) as a function of the target set size k, with $n = 128K$, $f = 2$, and 18 keys in total.

this by partitioning the population, n, into ν disjoint partitions of size n/ν. The space occupied by the $\log_2 \nu$ deleted keys per user is then used to increase the number of low level sets in each partition.

In this section we concentrate on larger, more realistic user population sizes. However, since each individual partition is small, we can apply the insight we have gained from our earlier small-population experiments.

Figures 10 and 11 compare the performance of the a single tree scheme for a population of 128K customers with the performance of schemes that employ the same number of keys (18) but with ν partitions. Within each partition we distribute the $\log \nu$ extra keys to achieve the lowest peak t; as we have seen before, this means that the extra keys are distributed among the lowest levels in the tree, thus adding key sets of sizes between 2 and 32. For each value of ν we ran the equivalent of the experiment we discussed in Section 6.3. We report only the results of the best (lowest peak t) extra-key distribution for each value of ν.

Figure 10 shows that the decrease in t is dramatic for a large range of target set sizes. In particular, the peak t drops by about 36%, from 24337 for a single partition to 15526 for $\nu = 1024$ partitions of size 128 each. Increasing the ν further reduces t for some values of k. However, for large target set sizes, and especially those with $k > n/2$, we pay a penalty in the number of transmissions. For such large target sets we have to use $t = \nu$ transmissions instead of one. We argue that as long as ν is substantially smaller than the peak t, the savings in t for smaller target sets far outweighs the penalty incurred for large target sets. Moreover, dealing with targets with $k > n/2$ can be done by maintaining a single additional broadcast key together with the partitions' keys.

Figure 11 shows that partitioning the users increases f_a for target sets with $k < n/2$. However, the peak f_a actually drops since we no longer use the very large key sets, e.g., those with size $n/2$ or $n/4$. Partitioning also improves the opportunity for $k \sim n/2$ (graph omitted).

We conclude that partitioning the users is an effective method for designing establishment key allocations. It is better to discard the large high-level key sets in favor of extra sets at the low levels. As a rule of thumb we suggest to use at least $\nu \approx \sqrt{n}$ partitions, and possibly more for larger values of n.

7 Dynamic Environments

In this section we discuss aspects of implementing our key management scheme in a dynamic environment such as the Internet. We distinguish between two types of dynamics. One type is the population's dynamics, i.e., the change in the population as new users are added and dropped from the MBone infrastructure. The other type is in-program dynamics, i.e., the case where paying customers join and leave a specific program broadcast while it is taking place.

7.1 Adapting to a Dynamic Population

Our first concern is how to adapt the static establishment key allocation we described before to a user population that is changing over time. We note that the user population is mainly growing as new networks are connected to the MBone. Networks depart from the MBone at a lower rate.

In Section 6.4 we showed that instead of building one monolithic establishment key allocation, it is better to split the population into partitions of a smaller size, say of 1024 users in each. Each of these partitions has its own establishment key allocation. Using this observation, we suggest building the establishment key allocation incrementally, in phases, as the population changes. A new partition is created at the beginning of each phase, with virtual "place holder" users. An establishment key allocation is constructed for the new partition, and each virtual user is assigned its keys. Each (real) new user that joins the MBone replaces a virtual user, and is assigned the virtual user's keys. The phase ends when all the virtual users in the new partition have been replaced by real users. At this point a new phase starts.

A user disconnecting from the MBone (not a temporary logoff) is marked as non-existing. Once the number of non-existing users in a partition drops below some threshold, say half the users are non-existing, the partition is deleted and all the users are rekeyed to a new partition. Note that the cost of rekeying a user is amortized over all the leave operations that required no rekeying in the past.

The virtual users in a new partition and the non-existing users in old partitions are all accounted for when key covers are computed for specific programs. However, their presence can only make f_a better: some of the redundant users that are part the cover $\mathcal{C}(K)$ (for which the ratio f is guaranteed) are not really there.

7.2 In-Program Dynamics

Next we discuss how decryption keys are transmitted in a dynamic environment, in which users join or leave a specific program, i.e., when the target set K is dynamically changing while the program is being transmitted. We believe that this type of fine-grained join/leave control is more appropriate in the MBone environment, where a single program may be quite lengthy.

To handle the dynamic changes in the target set, we divide the program's transmission time into slots of certain length, say 5 minutes. In each time slot, a different encryption key is used. We collect all the join/leave operations within slot $i-1$, compute the updated target set K_i, and recalculate the key cover $\mathcal{C}(K_i)$ for the next slot. The recipients in $R_{\mathcal{C}}(K_i)$ receive the new key.

Our goal is to quantify the effect of the in-program dynamics on the number of free-riders. To this end, we introduce the following definitions.

Definition 7.1 *Consider the first i slots of a program transmission.*

1. *Let $\mathcal{V}^i$ denote the set of users that can view all the i slots.*
2. *Let $\mathcal{P}^i$ denote the set of users that pay for all the i slots.*
3. *Let $\mathcal{N}^i$ denote the set of users that do not pay for any of the i slots.*

To measure the dynamic redundancy, we define the free-to-pay ratio $\rho(i)$, which is the proportion of non- and partial-paying users in the viewer set $\mathcal{V}^i$. Formally,

Definition 7.2 *Let* $\rho(i) = \frac{|\mathcal{V}^i \setminus \mathcal{P}^i|}{|\mathcal{V}^i|}$.

The ratio ρ is similar to a dynamic version of the actual redundancy f_a (recall Definition 2.6), but with a different denominator: $\rho(1) = (r_{\mathcal{C}}(K) - k)/r_{\mathcal{C}}(K)$ whereas $f_a = (r_{\mathcal{C}}(K) - k)/k$. For f-redundant establishment key allocations we have $\rho(1) \le 1 - 1/f$, however, this inequality may not hold for larger values of i since $\mathcal{V}^i$ and $\mathcal{P}^i$ evolve at different rates.

We define the dynamic opportunity, η_d, as the proportion of non-paying recipients (free-riders) in the non-paying population.

Definition 7.3 *Let* $\eta_d(i) = \frac{|\mathcal{V}^i \cap \mathcal{N}^i|}{|\mathcal{N}^i|}$.

This is a generalization of the opportunity η of Definition 2.7, and $\eta_d(1) = \eta$.

7.3 Experimenting with In-Program Dynamics

We conducted a series of simulation experiments to evaluate the effect of recalculating the key cover for every slot on the possibility to receive a program for free. We focused on relatively small user populations, since, as we have seen in Section 6.4, partitioning the population into many small partitions is advantageous.

The results we report here are all for $n = 1024$. We used 20 keys per user, using the best scheme we found in the experiments of Section 6.3, namely, distributing the 9 extra keys up to sets of size 8. We experimented with other values of n, and with other key distributions, with essentially the same results, so we omit the details.

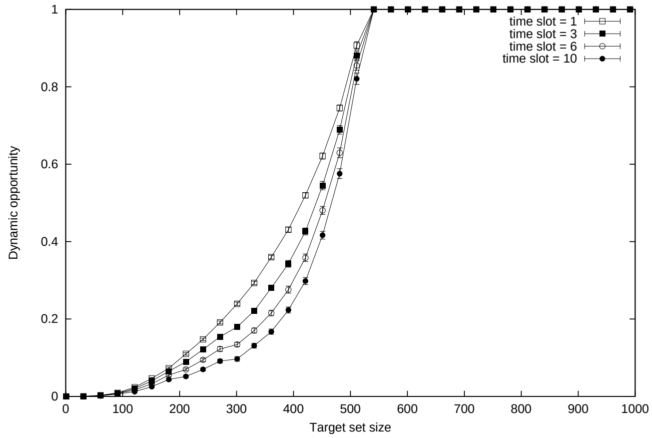

Figure 12: The dynamic opportunity $\eta_d(i)$ for $i = 1, 3, 6, 10$ slots, as a function of the target set size k, using $\tau = 0.01$ for $n = 1024$.

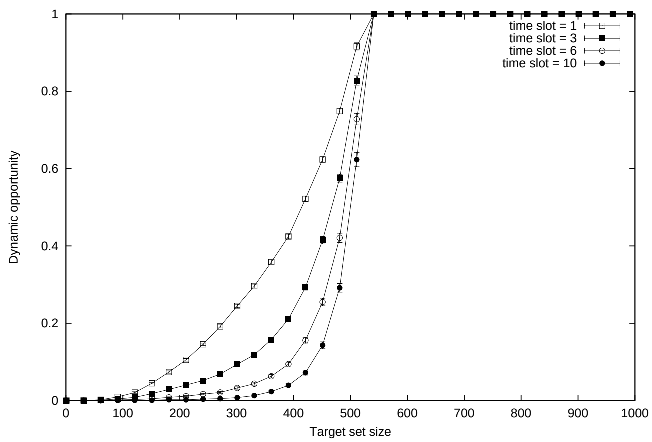

Figure 13: The dynamic opportunity $\eta_d(i)$ for $i = 1, 3, 6, 10$ slots, as a function of the target set size k, using $\tau = 0.1$ for $n = 1024$.

The in-program dynamics are captured by the following two parameters:

- $|K_1|$ is the initial paying population size;
- τ is the fraction of the paying population at slot i that stops paying during slot i (leaving users);

For simplicity we assume that every user that leaves in slot i is replaced by a joining user. Thus $|K_i| = |K_1|$ for all i. Exactly $\tau|K_{i-1}|$ leaving users are chosen at random from the set K_{i-1}, and the same number of joining users are chosen at random from $\mathcal{U} \setminus K_{i-1}$. The values we tested for τ were 0.01, 0.02, 0.05, and 0.1.

Figures 12 and 13 show the effect of the in-program dynamics on the dynamic opportunity $\eta_d(i)$. We see that even when the target set size changes by only 1% in each slot, the dynamic opportunity drops by between 33% and 52% for $250 \leq k \leq 450$ by the end of ten slots. When the target set size changes by 10% in each slot (Figure 13), the dynamic opportunity becomes negligible (less than 2%) after ten slots, for all target set sizes below 30% of the population. It is apparent from the figures that $\eta_d(i)$, as a function of k, tends to a step function when $i \to \infty$. The rate of the convergence to a step function depends on τ, with higher values of τ resulting in faster convergence.

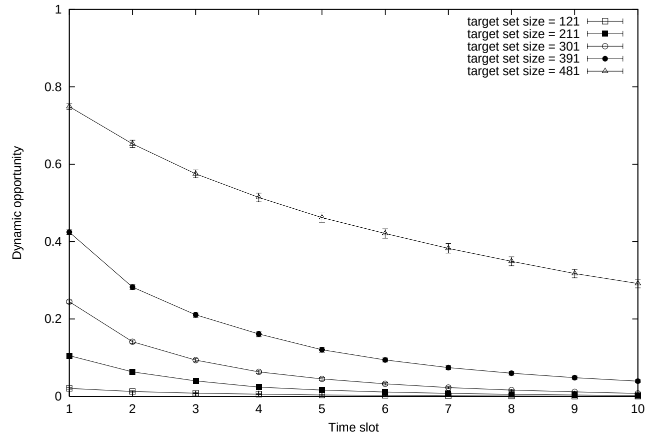

Figure 14: The dynamic opportunity $\eta_d(i)$ for several target set sizes, as a function of the slot number i, using $\tau = 0.1$ for $n = 1024$.

The explanation for this big gain, even for small change rates, lies in the algorithm we use to find the key cover (recall Figure 2). In step 3 of the algorithm, we pick a set that minimizes the actual redundancy. In many cases there are several choices for this set, and the arbitrary tie-breaking selection made by the algorithm determines the set of free-riders. A small random perturbation in the target set K randomizes the tie-breaking, resulting in a significant change in the set of free-riders.

Figure 14 shows the rates by which $\eta_d(i)$ drops as a function of time for $\tau = 10\%$. The dynamic opportunity drops to negligible levels for all but the largest target sets by the end of only 10 time slots. When the rate of change is 1% (graph omitted) the decrease is slower, however, as we said in the discussion of Figure 12, the drop is still substantial in the mid-sized target sets.

Figure 15 shows the dramatic drop in the dynamic opportunity when the rate of change τ grows from 1% to 10%, in comparison to the opportunity η for a static target set. We see that even when the dynamics are minimal ($\tau = 0.01$) there is a 60% drop in the opportunity, e.g., from $\eta = 0.24$ to $\eta_d(10) = 0.097$ for $k = 301$. For higher rates of change the drop is even more pronounced.

We conclude from all the above discussion that the in-program dynamics makes the service provider's situation more favorable. The changing target population results in significantly better performance from our f-redundant establishment key allocations: the free-rider's opportunity rapidly decreases, even for low change rates. Thus it becomes increasingly hard to be able to watch an entire program for free, as the number of slots increases.

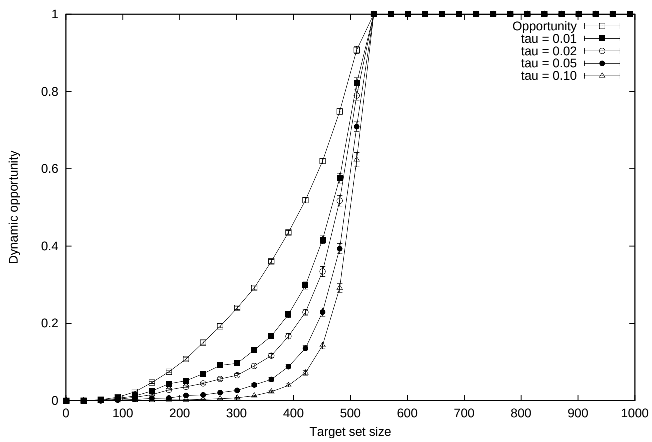

Figure 15: The dynamic opportunity $\eta_d(10)$, after 10 slots, for different values of τ, as a function of the target set size k, for $n = 1024$.

8 Conclusions

We have demonstrated that by allowing a controlled number of free-riders we are able to design establishment key allocations that meet the hard limitations placed on secure key storage by current technology. We do this while addressing the ambitious goal of allowing *every* possible subset of users to be a target set (rather than only sets of a small fixed cardinality). We showed that despite these constraints, our schemes use substantially fewer transmissions than the naive designs. Moreover, although our schemes guarantee that the ratio between the numbers of free-riders and intended receivers is at most $f-1$, the achieved redundancy ratio f_a is typically much better than the guarantee. These desirable properties are enhanced even further in dynamic, MBone-like, environments: the opportunity for a free ride quickly decreases when the paying user population is dynamic.

We have also identified some general design principles for such systems. We found that adding extra establishment key sets helps, provided that they are added at the low levels. We also found that partitioning the population into many small partitions is more effective than handling the whole population at once, since by eliminating the very large key sets we can add extra keys in each partition without exceeding the key storage limitations. We conclude that our schemes are quite practical for applications where some free-riders may be tolerated.

We believe that more can be done in this area. It would be useful to provide a model for the analysis of establishment key allocation schemes. One would also like to model the user behavior and its willingness to pay per service that might be supplied for free. Finally, the work on dynamic behavior in the Internet require a more rigorous study.

References

[1] C. Blundo and A. Cresti. Space requirements for broadcast encryption. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT'94, LNCS 950*, pages 287–298. Springer-Verlag, 1994.

[2] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Generalized Beimel-Chor schemes for broadcast encryption and interactive key distribution. *Theoretical Computer Science*, 200(1–2):313–334, 1998.

[3] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO'94, LNCS 839*, pages 257–270. Springer-Verlag, 1994.

[4] J. L. Cohen, M. H. Etzel, D. W. Faucher, and D. N. Heer. Security for broadband digital networks. *Communications Technology*, pages 58–69, August 1995.

[5] C. J. Colbourn and J. H. Dinitz. *The CRC Handbook of Combinatorial Designs.* CRC Press, Boca Raton, 1996.

[6] H. Eriksson. MBone: the multicast backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[7] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, July 1998.

[8] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology – CRYPTO'93, LNCS 773*, pages 480–491. Springer-Verlag, 1994.

[9] E. Gabber and A. Wool. On location-restricted services. *IEEE Networks Magazine*, 13(6):44–52, November/December 1999.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[11] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Euro. J. Operational Research*, 101(1):81–92, August 1997.

[12] D. S. Hochbaum. (ed.) *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, Boston, MA, 1995.

[13] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Computer System Sci.*, 9:256–278, 1974.

[14] D. Kravitz and D. Goldschlag. Conditional access concepts and principles. In M. Franklin, editor, *Proc. Financial Cryptography'99, LNCS 1648*, Anguilla, BWI, February 1999. Springer-Verlag.

[15] A. M. Law and W. D. Kelton. *Simulation Modelling and Analysis.* McGraw-Hill, 2nd edition, 1991.

[16] L. Lovász. On the ratio of optimal integral and fractional covers. *Disc. Math.*, 13:383–390, 1975.

[17] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98, LNCS 1403*, pages 512–526, Espoo, Finland, 1998. Springer-Verlag.

[18] B. M. Macq and J.-J. Quisquater. Cryptology for digital TV broadcasting. *Proceedings of the IEEE*, 83(6):944–957, 1995.

[19] J. McCormac. *European Scrambling Systems 5*. Waterford University Press, Waterford, Ireland, 1996.

[20] W. H. Mills and R. C. Mullin. Coverings and packings. In J. H. Dinitz and D. R. Stinson, editors, *Contemporary Design Theory: A Collection of Surveys*, pages 317–399. John Wiley & Sons, 1992.

[21] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proc. ACM SIGCOMM*, September 1997.

[22] M. Naor and B. Pinkas. Threshold traitor tracing. In *Advances in Cryptology – CRYPTO'98, LNCS 1462*. Springer-Verlag, 1998.

[23] J. Schönheim. On coverings. *Pacific J. Math.*, 14:1405–1411, 1964.

[24] D. R. Stinson and T. van Trung. Some new results on key distribution patterns and broadcast encryption. *Designs, Codes and Cryptography*, 14(3):261–279, 1998.

[25] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. Internet Draft, September 1998. Available from `http://www.ietf.org/ID.html`.

[26] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[27] A. Wool. Key management for encrypted broadcast. In *Proc. 5th ACM Conf. Computer and Communications Security (CCS)*, pages 7–16, San Francisco, November 1998. To appear in ACM Trans. Information and System Security.

Michel Abdalla Michel Abdalla (S'94) was born in Rio de Janeiro, Brazil. He received the B.Sc. degree in Electronics Engineering and the M.Sc. degree in Electrical Engineering from Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, in 1993 and 1996, respectively. From 1993 to 1996, he has worked with the Computer Network Research Group at Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil. Since 1997, he has been a member of the Cryptography and Security Laboratory in the Computer Science and Engineering Department at the University of California, San Diego, La Jolla, California, where he is presently a Ph.D. candidate. His current research interests span several areas in cryptography including encryption, digital signatures, threshold cryptosystems and key management. He is a member of the IEEE and IACR.

Yuval Shavitt Yuval Shavitt (IEEE S'88–M'97) received the B. Sc. in Computer Engineering (cum laude), M. Sc. in Electrical Engineering and D. Sc. from the Technion — Israel Institute of Technology, Haifa, Israel in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces first as a system engineer and the last two years as a software engineering team leader. He spent the summer of 1992 as summer student at IBM T. J. Watson Research Center. After graduation he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Since 1997 he is a Member of Technical Stuff at Bell Labs, Lucent Technologies, Holmdel, NJ. His recent research focuses on active networks and their use in network management, QoS routing, and Internet mapping and characterization.

Avishai Wool Avishai Wool (IEEE Computer Soc.'97 / ACM'96) received a B.Sc. in Mathematics and Computer Science from the Tel Aviv University, Israel, in 1989, and an M.Sc. and Ph.D. in Computer Science from the Weizmann Institute of Science, Israel, in 1992 and 1996, respectively. He then joined Bell Laboratories, where he is a member of technical staff in the Secure Systems Research Department. His research interests include computer and network security, firewalls, distributed computing, quorum systems, and fast communication networks.