Self-Healing Key Distribution with Revocation

Jessica Staddon
Palo Alto Research Center
staddon@parc.com

Sara Miner[*]
U. C. San Diego
sminer@cs.ucsd.edu

Matt Franklin
U. C. Davis
franklin@cs.ucdavis.edu

Dirk Balfanz
Palo Alto Research Center
balfanz@parc.com

Michael Malkin[†]
Stanford University
mikeym@stanford.edu

Drew Dean[‡]
SRI International
ddean@csl.sri.com

May 28, 2002

[*]The majority of this work was completed while the author was a summer intern at Xerox PARC.
[†]The majority of this work was completed while the author was a summer intern at Xerox PARC.
[‡]The majority of this work was completed while the author was employed by Xerox PARC.

Self-Healing Key Distribution with Revocation

Jessica Staddon
Palo Alto Research Center
staddon@parc.com

Sara Miner[*]
U. C. San Diego
sminer@cs.ucsd.edu

Matt Franklin
U. C. Davis
franklin@cs.ucdavis.edu

Dirk Balfanz
Palo Alto Research Center
balfanz@parc.com

Michael Malkin[†]
Stanford University
mikeym@stanford.edu

Drew Dean[‡]
SRI International
ddean@csl.sri.com

Abstract

We address the problem of establishing a group key amongst a dynamic group of users over an unreliable, or lossy, network. We term our key distribution mechanisms self-healing *because users are capable of recovering lost group keys on their own, without requesting additional transmissions from the group manager, thus cutting back on network traffic, decreasing the load on the group manager, and reducing the risk of user exposure through traffic analysis. A user must be a member both before and after the session in which a particular key is sent in order to be able to recover the key through self-healing. Binding the ability to recover keys to membership status enables the group manager to use short broadcasts to establish group keys, independent of the group size. In addition, the self-healing approach to key distribution is stateless, meaning that a group member who has been off-line for some time is able to recover new session keys immediately after coming back on-line.*

1. Introduction

One method for enabling secure multicast communication is the periodic distribution of a new key (called a *session* key) to group members. All messages exchanged within the group during a fixed interval of time, or session, are communicated securely through encryption under this session key. We assume that, prior to the start of each session, the group manager broadcasts a packet containing that session's key (and perhaps, other information) to the group. Because group membership is dynamic (that is, users may be added and removed from the group periodically), the key distribution broadcast targets only current group members.

The problem of distributing keys over a *reliable* channel has received much attention (see, for example, [12, 20, 33, 39]). In this paper, we study a pragmatic variant of this problem that has received much less attention–namely, how to distribute session keys in a manner that is resistant to packet loss.

In an unreliable network, the key distribution broadcast for a particular session might never reach a user. Requiring that each such user contact the group manager to request a re-transmission would contribute to the traffic on a network that might already be heavily burdened, and, when group size is large, such re-transmissions could potentially overwhelm the group manager. Furthermore, in some high-security environments (*e.g.,* military applications) it can be important that users avoid sending all but essential messages, lest they make themselves vulnerable by revealing their location. Hence, we propose a solution that is noninteractive.

SELF-HEALING KEY DISTRIBUTION. The central concept of this paper is a type of noninteractive key distribution that we call *self-healing*. The idea is that group members who (due to network packet loss) do not receive a particular session key via the key distribution broadcast, can recover the session key *on their own*. A group member accomplishes this by combining information from any key distribution broadcast preceding the lost packet with information from any key distribution broadcast following it. In other words, in order to recover a lost session key, the user must have received key distribution broadcasts for any two sessions which "sandwich" the session corresponding to the lost key distribution broadcast. Hence, when self-healing key distribution is implemented for a sequence of m sessions, it is possible to miss all but the first and last key distribution

[*]The majority of this work was completed while the author was a summer intern at Xerox PARC.
[†]The majority of this work was completed while the author was a summer intern at Xerox PARC.
[‡]The majority of this work was completed while the author was employed by Xerox PARC.

broadcasts, and still be able to recover all the session keys.

Basing session key recovery on the possession of sandwiching broadcasts allows us to use a flat, rather than hierarchical, key management system. In such a system, each personal key[1] is known to exactly one user (enabling traceability) and broadcasts are constructed in a stateless manner. The cost of these benefits is an increase in communication overhead. However, because the keying information is naturally decoupled from the content in the session key setting, the overhead is incurred on the smaller payload, the session keys, and so is quite reasonable. On the content, a low-overhead reliability mechanism (*e.g.,* forward error correction) can be used.

Our key distribution schemes not only provide self-healing, but the ability to revoke users from, and add users to, the group, while being resistant to collusion attacks. If a key distribution mechanism cannot be broken by any coalition of up to t users, we say it is resistant to coalitions of size t.

APPLICATIONS. Self-healing key distribution appears to be quite useful in high-security operations (such as the military), where it is necessary to change session keys frequently and to be able to revoke users quickly. Self-healing key distribution works well here because the length of time over which a user must buffer encrypted messages is short, and revocation can be accomplished quickly with the broadcast of a single packet. In addition, the self-healing approach may be useful in commercial content distribution applications in which the content is highly sensitive. For example, during mergers and acquisitions extensive negotiations involving many representatives from both sides may take place. Frequent session key changes may be necessary and the ability to revoke low-ranking parties during certain exchanges is desirable. We emphasize that in any application of self-healing key distribution the expected number of consecutive sessions in which key distribution packets are lost must be less than the number of sessions in between any two intervals of membership for a particular user. This appears to often be the case. For example, in group conferencing over the Internet, a burst of loss amongst the key distribution packets is likely to only cover an interval of time on the order of seconds, however the length of time during which a user may be revoked (to allow for discussion of sensitive information, for example) will be at least on the order of several minutes. The self-healing approach to reliable key distribution is quite appropriate for such applications because it is unlikely that a user will abuse self-healing by leaving and rejoining the group within a short time period.

RESULTS. For applications such as those described above, we show that with simple, polynomial-based secret sharing techniques, it is possible to achieve noninteractive resistance to packet loss through small broadcasts. In particular, we provide an unconditionally secure construction with broadcast overhead that is on the order of $(t^2m)\log q$ bits, where $\log q$ is the session key size, t is the collusion resistance, and m denotes the number of sessions over which self-healing is possible (which is closely correlated with anticipated packet loss). Further, we show that it is possible to achieve broadcasts of size $O((t^2 + mt)\log q)$ bits, by shifting a moderate amount of computation to the user's end. Each of these constructions provides for *fast* self-healing (the core operation is simple polynomial interpolation) over a fixed set of m sessions and is resistant to collusion. We discuss how to use modular exponentiation-based secret sharing [16] to extend the lifetime of these constructions by allowing users to *evolve* their personal keys from a base set to an appropriate set of keys for the current set of sessions. In all of these constructions, recovery from loss is possible with no delay on the user's part–after several key distribution packets are lost, a single received key distribution packet is sufficient to recover all the missed session keys. The constructions are stateless; group members aren't penalized for being off-line for a period of time. This property is important in wireless applications in which members can quickly become off-line by moving out of broadcast range. In addition, all of the personal keys in the system are traceable. A consequence of the traceability and collusion resistance is that the only way to break the system in a long-term sense without risk of identification, is to form a coalition of more than t users.

As part of our work, we introduce a new general technique for distributing unique keys to a select subset of users. This result is of independent interest and is a useful extension of earlier techniques for distributing a *common* key to a select subset of users [29].

Finally, we discuss the practical issues that must be addressed when implementing a self-healing key distribution scheme. The core issue is parameter choices that are both appropriate for the intended application and compatible with existing network protocols. We illustrate the tradeoffs between the system parameters that exist while staying within IP packet size constraints. Even if the parameters are such that packet fragmentation is required (*i.e.,* the size constraints aren't met) the fragments can be formed in such a way that each is useful to a member whether or not any other fragments are received. As a result, a member may still be able to use the received packets to self-heal, or recover session keys directly, even when the packets are fragments of the actual key distribution broadcasts.

OVERVIEW. The rest of the paper is organized as follows. In Section 1.1 we provide an overview of earlier work in the areas of key distribution and multicast. Section 2 defines most of the necessary terminology. Section 3 introduces

[1]The *personal key* is the collection of secrets that allows users to decrypt broadcast messages.

the new techniques of this paper: the self-healing mechanism and the revocation mechanism. Section 4 discusses our self-healing session key distribution scheme. In Section 5 we show how to reduce the broadcast size by shifting some computation to the user's end. Section 6 describes how to extend the lifetime of our approach. Section 7 discusses the practical issues encountered when implementing this work, and Section 8 concludes the paper. Background on the information theory concepts used and some security proofs and lower bounds can be found in the appendices.

1.1. Related Work

In [41, 31] two key distribution protocols that are resistant to packet loss through noninteractive means are described. Both are motivated by the single sender content distribution setting and take a tree-based approach to key distribution and achieve resistance to loss by appending additional key update information to the packets that *follow* a key distribution broadcast. In [41], resistance to packet loss is ensured by using error-correcting codes to generate information about past group keys. If a certain fixed number of packets is received after a lost one, it is possible for the user to reconstruct the lost information. In [31], these ideas are built upon to allow resistance to correlated packet loss. Depending on the membership change(s) that trigger the rekeying, "hints" for updated keys are attached to subsequent data packets. The hints can be as small as half the size of the keys themselves, but leave the user with significant work to do in order to recover missed keys.

When redundancy is only used *after* each key distribution packet as in [41, 31], care must be taken to ensure that new group members don't receive enough packets to allow recovery of keys to which they are not entitled. Specialized communication (perhaps including unicast) is thus necessary to control what members receive. Achieving small broadcast size with such specialized communication would seem to necessitate the hierarchical key management systems from [39, 40], that are used in [41, 31]. One shortcoming of using this approach is that such systems are stateful, that is, as the keys in the tree are updated, an off-line member quickly becomes shut out of the group and cannot rejoin without assistance from the group manager. In addition, in a hierarchical key management system many keys are known to *subsets* of users, and giving away such keys allows outsiders to enter the system in a way that is difficult to trace. The approaches in [41, 31] differ significantly from the approach taken here because the self-healing property requires that *any* pair of preceding and following packets be sufficient for recovering the lost key. With this self-healing requirement, it is possible to communicate with all group members through short broadcasts even though the underlying set of personal keys is flat rather than hierarchical (each key is stored by one user), which has the advantage of permitting traceability. In addition, the flat key structure doesn't penalize members for being off-line for a period of time. We note also that in [31] the keying information is not decoupled from the content. Pairing the two makes sense if the group manager is the only sender, but in the multi-sender setting that we consider, doing so would require passing all messages through the group manager first, as appending the necessary keying information in a secure way requires knowledge of the users' personal keys.

Key distribution is at the core of many multicast and broadcast encryption [5, 17] schemes. Our constructions rest on a new technique for distributing distinct keys that is an extension of techniques for distributing common keys to subsets of users due to Naor and Pinkas [29]. In addition, our approach to the multicast problem is similar to the one taken by Kronos [33], in that we also use periodic rekeying. For other multicast and broadcast encryption techniques see [11, 17, 20, 21, 22, 28, 32, 37]. Graph-based multicast constructions are given in [39, 40, 26], and a method for reducing the number of update messages needed by a previously off-line member in such schemes is given in [30]. Lower bounds for communication and storage in multicast schemes are proven in [12].

Because our goal is secure communication for large groups, we take a broadcast-based approach to key distribution. There are many other scenarios in which key distribution is needed. We briefly mention some of them for completeness. Initially, key distribution was mostly studied with the goal of establishing a shared secret between two parties [7]. A generalization of the problem studied in [7], that of establishing a shared key amongst a group of any size, is studied in [23, 19] in roughly the same model. A Diffie-Hellman based solution for authenticated key distribution is given in [43]. The two party key distribution problem is studied in the computational setting in [14, 24, 42]. In [36, 9, 2, 8], interactive key distribution is studied. The formal analysis of key distribution protocols is considered in [18, 38]. Provably secure two party key distribution with active adversaries is studied in [3, 4, 35, 6].

2. Definitions and Notation

In a key distribution scheme, a group manager seeks to establish a new *unique* key with each user over a broadcast channel (See Appendix C for a formal definition). In a *session* key distribution scheme, a group manager seeks to establish a common key (the session key) with everyone in the group at the beginning of each session, where a session is simply a fixed interval of time. In each setting, the ability to revoke users, and thus prevent them from learning new keys, is important. We say a scheme has t-revocation capability if it is possible to prevent t users at a time from learning the new session key. When distributing session keys, we also consider the self-healing property, which states that a member in three sequential (though not necessarily consecutive) sessions can recover the session key corresponding to the intermediate session by using information recovered from the first and last of the three broadcasts. All of our schemes are resistant to coalitions of t users. That is, any t colluding users (whether revoked or not) are unable to recover information they are not entitled to.

We begin by defining the notation of this paper and the unconditionally secure model of session key distribution. Many of our definitions and results make use of information theory concepts such as the entropy function, $H(\cdot)$. A brief review of the necessary concepts is in Appendix A.

We consider a setting in which there is a group manager and n users $U_1, \ldots, U_n$. All of our operations take place in a finite field, F_q, where q is a prime that is larger than n. Each user, U_i, stores a personal key, $S_i \subseteq F_q$ (*i.e.,* S_i may be a subset of elements of F_q). We use k to denote a single key (*i.e.,* an element of F_q). We allow for the possibility that individual keys may be related.

Definition 1 *[Key independence] $\{k_i\}_{i\in\{1,\ldots,n\}} \subseteq F_q$ is a set of t-wise independent keys, if for every subset of t distinct indices $\{i_1, \ldots i_t\}$, $H(k_{i_1}|k_{i_2}, \ldots k_{i_t}) = H(k_{i_1})$.*

We denote the number of sessions by m, and the set of users who are revoked in session j, and thus unable to recover that session's key, by R. If $U_i \notin R$, we say U_i is a *member* (or, an *active* user). The session keys $\{K_1, \ldots, K_m\}$, are generated independently at random. For $j \in \{1, \ldots, m\}$, the session key, K_j, is sent to the group members through a broadcast, $\mathcal{B}_j$, from the group manager. For any non-revoked user U_i, the jth session key, K_j, is determined by $\mathcal{B}_j$ and S_i (the set of revoked users, R, will be clear from context).

Because in a session key distribution scheme a user potentially learns from $\mathcal{B}_j$, information about session keys other than K_j, it is helpful to introduce a variable $z_{i,j}$ to represent all the information U_i learns through knowledge of *both* $\mathcal{B}_j$ and S_i. More precisely: $H(z_{i,j}|\mathcal{B}_j, S_i) = 0$ but $H(z_{i,j}|\mathcal{B}_j,) = H(z_{i,j}) = H(z_{i,j}|S_i)$. For example, if U_i is a group member, then $z_{i,j}$ will include K_j and possibly information on other session keys, whereas if U_i is revoked then $z_{i,j}$ contains no information on K_j and may in fact be the empty set.

We emphasize that it is important to prepare for all types of collusion attacks when designing key distribution schemes. If the scheme is such that sensitive information is embedded in users' personal keys (*e.g.,* [15]) a coalition of users may be unwilling to share their personal keys and consequently can only attack session keys. Such a coalition could consist of α revoked users who collude with $t - \alpha$ new group members to recover session keys for sessions in which none of the colluding users were members. Security against such a collusion attack motivates our definition of self-healing in Definition 2.

Definition 2 *[Session Key Distribution]*
Let $t, i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.

1. *$\mathcal{D}$ is a session key distribution scheme if the following are true:*
   (a) *For any member U_i, K_j is determined by $z_{i,j}$, which in turn is determined by $\mathcal{B}_j$ and S_i ($H(K_j|z_{i,j}) = 0$ and $H(z_{i,j}|\mathcal{B}_j, S_i) = 0$).*
   (b) *For any set $B \subseteq \{U_1, \ldots, U_n\}$, $|B| \le t$, and $U_i \notin B$, the users in B cannot determine anything about S_i ($H(S_i|\{S_{i'}\}_{U_{i'}\in B}, \mathcal{B}_1, \ldots, \mathcal{B}_m) = H(S_i)$).*
   (c) *What members $U_1, \ldots, U_n$ learn from $\mathcal{B}_j$ can't be determined from the broadcasts or personal keys alone ($H(z_{i,j}|\mathcal{B}_1, \ldots, \mathcal{B}_m) = H(z_{i,j}) = H(z_{i,j}|S_1, \ldots, S_n)$).*

2. *$\mathcal{D}$ has t-revocation capability if given any set $R \subseteq \{U_1, \ldots, U_n\}$ where $|R| \le t$, the group manager can generate a broadcast $\mathcal{B}_j$, such that for all $U_i \notin R$, U_i can recover K_j ($H(K_j|\mathcal{B}_j, S_i) = 0$), but the revoked users cannot ($H(K_j|\mathcal{B}_j, \{S_{i'}\}_{U_{i'}\in R}) = H(K_j)$).*

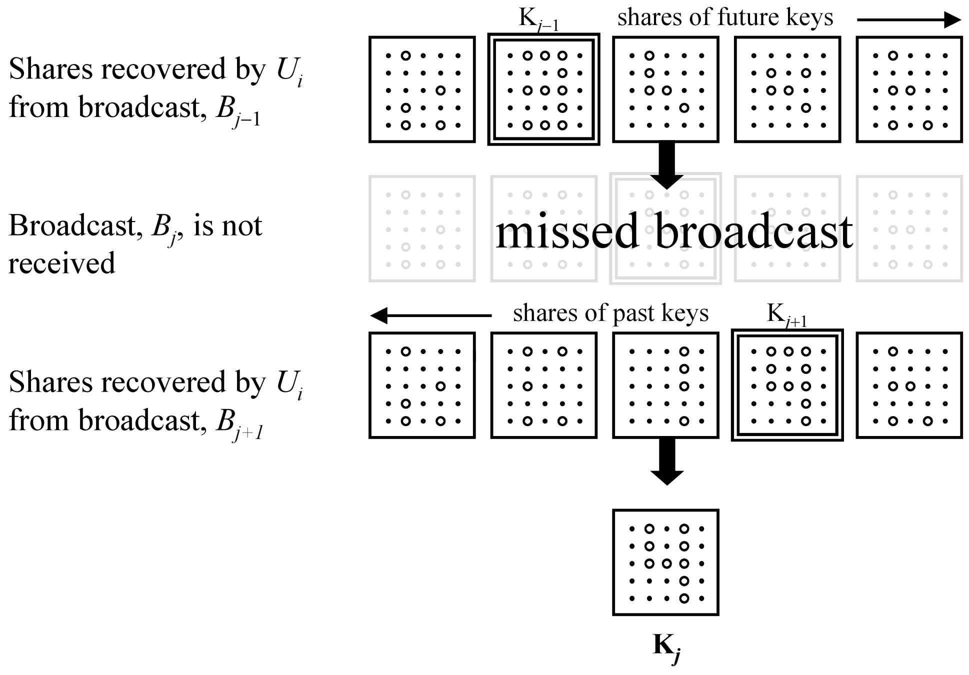
3. *$\mathcal{D}$ is self-healing if the following are true for any $1 \le j_1 < j < j_2 \le m$:*
   (a) *For any U_i who is a member in sessions j_1 and j_2, K_j is determined by the set, $\{z_{i,j_1}, z_{i,j_2}\}$ ($H(K_j|z_{i,j_1}, z_{i,j_2}) = 0$).*
   (b) *For any disjoint subsets $B, C \subset \{U_1, \ldots, U_n\}$ where $|B \cup C| \le t$, the set $\{z_{i',j}\}_{U_{i'}\in B, 1\le j\le j_1} \cup \{z_{i',j}\}_{U_{i'}\in C, m\ge j\ge j_2}$, contains no information on K_j ($H(K_j|\{z_{i',j}\}_{U_{i'}\in B, 1\le j\le j_1} \cup \{z_{i',j}\}_{U_{i'}\in C, m\ge j\ge j_2}) = H(K_j)$).*

We consider *computationally* secure session key distribution in Section 6. Because the definition of session key distribution in the computational setting is a natural variation on the above definition, we do not include it here, but provide it instead in Appendix E.

3. New Techniques

3.1. Self-Healing

The idea behind this technique is to use secret sharing [34], to bind the ability of users to recover from packet loss to the user's membership status. From each broadcast, a



Shares recovered by U_i from broadcast, B_{j-1}
K_{j-1}
shares of future keys
Broadcast, B_j, is not received
missed broadcast
shares of past keys
K_{j+1}
Shares recovered by U_i from broadcast, B_{j+1}
K_j

Figure 1. The self-healing mechanism. From $\mathcal{B}_{j-1}$, U_i recovers a share of K_{j-2}, the key K_{j-1} and shares for K_j, K_{j+1} and K_{j+2}. From $\mathcal{B}_{j+1}$, U_i recovers the same share of K_{j-2}, new shares of K_{j-1} and K_j, the key K_{j+1} and the same share of K_{j+2}. As a result of $\mathcal{B}_{j-1}$ and $\mathcal{B}_{j+1}$, U_i now has complementary shares of K_j and can recover $K_j = 4$, by self-healing, even though broadcast $\mathcal{B}_j$ wasn't received.

user recovers the current session key and a *share* of each of the previous and future session keys. Hence, in each broadcast, a user learns either the actual key or a share of the actual key for each of the m sessions. The share of K_j that is received in each session $j_1 < j$ is complementary to the share of K_j that is received in each session $j_2 > j$. Hence, a user who is a member in both session j_1 and j_2 will be able to reconstruct K_j, even if $\mathcal{B}_j$ isn't received. Figure 1 represents the self-healing property in an intuitive way; the value of the session key is ambiguous when each share is considered alone, but when the shares are combined, the value of K_j becomes clear.

In order to provide resistance to collusion attacks, in the self-healing key distribution schemes that are based on this mechanism, the shares recovered by different users are different. The collusion resistance of a key distribution scheme is correlated with the degree of dependence between the shares recovered by the users in each period. We can accomplish any desired level of coalition resistance by using polynomials of sufficiently high degree to determine the values of the shares.

Construction 1 *A self-healing session key distribution scheme (without revocation capability)*

1. *(Set-up) Let t be a positive integer. The group manager chooses $2m$ polynomials in $F_q[x]$, each of degree t, $h_1, \ldots, h_m, p_1, \ldots, p_m$, and m session keys, $K_1, \ldots, K_m \in F_q$, all at random. For each $j \in \{1, \ldots, m\}$, define a polynomial in $F_q[x]$, $q_j(x) = K_j - p_j(x)$. For $i \in \{1, \ldots, n\}$, user U_i stores the personal key $S_i = \{i, h_1(i), \ldots, h_m(i)\} \subseteq F_q$.*

2. *(Broadcast) In session $j \in \{1, \ldots, m\}$, the broadcast is:*

$$\begin{aligned}\mathcal{B}_j \quad = \quad & \{h_1(x) + p_1(x), \ldots, h_{j-1}(x) + p_{j-1}(x),\\ & h_j(x) + K_j,\\ & h_{j+1}(x) + q_{j+1}(x), \ldots, h_m(x) + q_m(x)\}\end{aligned}$$

3. *(Session Key and Shares Recovery in Session j) For all $i \in \{1, \ldots, n\}$, U_i recovers K_j from broadcast $\mathcal{B}_j$ by evaluating $h_j(x) + K_j$ at i and subtracting $h_j(i)$ (the latter is part of S_i). Similarly, U_i recovers session key shares $\{p_1(i), \ldots, p_{j-1}(i), q_{j+1}(i), \ldots, q_m(i)\}$. Self-healing is then possible because in session $j_1 < j$, U_i recovers share $q_j(i)$ and in session $j_2 > j$, U_i recovers share $p_j(i)$, and $p_j(i) + q_j(i) = K_j$.*

Adding a user to this scheme during session j' is straightforward, provided the underlying field is sufficiently large. The group manager sends a new member a unique identity, $i \in F_q$, and the corresponding points on the polynomials $\{h_j(i)\}_{j \in \{j', \ldots, m\}}$. However, Construction 1 has no revocation capability. In Section 4 we describe how the construction may be combined with Construction 2 to achieve self-healing key distribution with revocation.

We prove the security of Construction 1 (Lemma 1) in Appendix B.

Lemma 1 *Construction 1 is an unconditionally secure, self-healing, session key distribution scheme (with no revocation capability).*

User storage and broadcast size in Construction 1 are both essentially optimal, as shown by the following lemmas. The proofs are in Appendix D.

Lemma 2 *In an unconditionally secure, session key distribution scheme, if user U_i is entitled to all m session keys, then $H(S_i) \geq m \log q$, for each $i \in \{1, \ldots, n\}$.*

Lemma 3 *In an unconditionally secure, self-healing session key distribution scheme, $H(\mathcal{B}_j)$ is $\Omega(mt) \log q$.*

3.2. Revocation

In this section we describe a mechanism for distributing one set of distinct (but related) keys to a select subset of users over a broadcast channel. Later, this mechanism will allow us to add revocation capability to the self-healing technique of Section 3.1. Note that the ability to distribute *distinct* keys to a subset of users is essential to self-healing key distribution, because although our main goal is the distribution of common keys (*i.e.,* session keys), we do this reliably by also distributing *shares* of keys, and these shares must be distinct to ensure collusion resistance. The mechanism we present here can be viewed as a generalization of the Naor-Pinkas unconditionally secure method for establishing a *common* key over a broadcast channel, [29].

The keys distributed in this mechanism are each a point on a polynomial. The size of the broadcast grows with the square of the degree of collusion resistance desired, not with the total number of users. Figure 2 illustrates the goals of the key distribution mechanism.

Construction 2 *A key distribution scheme with t-revocation capability (and without self-healing)*

1. *(Set-up) Let t be a positive integer. Let $N \in F_q$, be an element that is not equal to any user's index. The group manager chooses at random from $F_q[x, y]$ a polynomial, $s(x, y) = a_{0,0} + a_{1,0}x + a_{0,1}y + \ldots + a_{t,t}x^t y^t$. For $i = 1, \ldots, n$, user U_i stores the personal key, $(N, i, s(i, i))$.*
2. *(Broadcast) The group manager chooses at random a polynomial of degree t in $F_q[x]$, $f(x)$. Let $W \subseteq \{1, \ldots, n\}$, $|W| = t$, consist of the indices of the users that should not be allowed to recover a new key from the broadcast. The broadcast consists of the following polynomials:*

$$\{f(x) + s(N, x)\} \cup \{w, s(w, x) : w \in W\}$$

3. *(Key Recovery) A user U_i such that $i \notin W$, can evaluate each polynomial $s(w, x)$ at $x = i$ to get t points on the polynomial $s(x, i)$. Coupling these with his personal key $s(i, i)$, U_i has $t + 1$ points on $s(x, i)$ and so is able to recover that polynomial and evaluate it at $x = N$ to recover $s(N, i)$. U_i may then evaluate $(f(x) + s(N, x))$ at $x = i$, subtract off $s(N, i)$ and recover a new individual key, $f(i)$.*

Because this technique is of independent interest, we demonstrate its security before it is combined with the self-healing mechanism. The proof of the following lemma may be found in Appendix C.

Lemma 4 *Construction 2 is an unconditionally secure key distribution scheme with t-revocation capability.*

Note that the keys distributed in Construction 2, $\{f(1), \ldots, f(n)\}$ are $(t + 1)$-wise independent because $f(x)$ is of degree t. The size of the broadcast, $\mathcal{B}$, in Construction 2 is $O(t^2 \log q)$. The Naor-Pinkas scheme, which is an unconditionally secure method of distributing a common key, has broadcast size $O(t \log q)$, so moving from the distribution of a single key to the distribution of a set of (t+1)-wise independent keys has multiplied the broadcast length by t.

4. Self-Healing Session Key Distribution

By combining the techniques of Sections 3.1 and 3.2, we construct a session key distribution scheme that has t-revocation capability and is self-healing.

Construction 3 *Unconditionally secure self-healing session key distribution*

1. *(Set-up) Let t be a positive integer, and let N be an element of F_q that is not equal to any user index. The group manager chooses m polynomials $p_1(x), \ldots, p_m(x)$ in $F_q[x]$, each of degree t, and m session keys $K_1, \ldots, K_m \in F_q$, all at random, and defines a polynomial, $q_j(x) = K_j - p_j(x)$, for each $j = 1, \ldots, m$. For each $j \in \{1, \ldots, m\}$, the group manager chooses m polynomials in $F_q[x, y]$ at random, $s_{1,j}, \ldots, s_{m,j}$, where for $i = 1, \ldots, m$,*

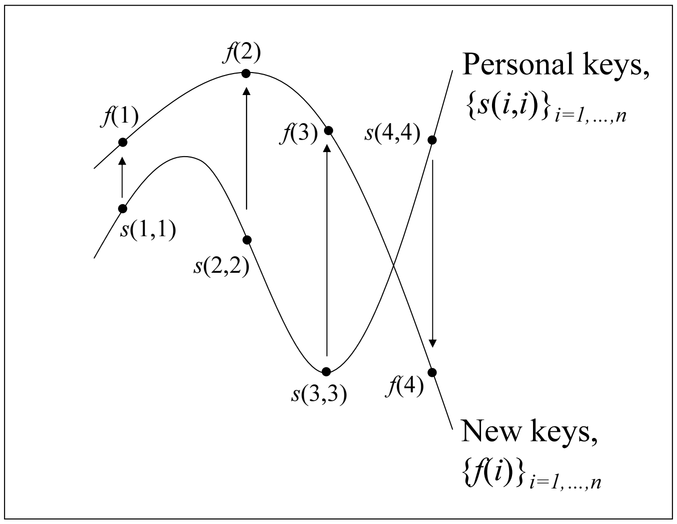

f(1)
f(2)
f(3)
s(4,4)
Personal keys, $\{s(i,i)\}_{i=1,\dots,n}$
s(1,1)
s(2,2)
s(3,3)
f(4)
New keys, $\{f(i)\}_{i=1,\dots,n}$

Figure 2. The Revocation Mechanism. For $i = 1, \dots, n$, U_i stores personal key $(N, i, s(i,i))$. After the broadcast, a member U_i is able to recover a new key $f(i)$, but learns nothing about $f(j)$ for $j \neq i$. A revoked user, $U_{i'}$, learns nothing about any of the new keys, $\{f(i)\}_{i\in\{1,\dots,n\}}$.

$s_{i,j}(x,y) = a^{i,j}_{0,0} + a^{i,j}_{1,0}x + a^{i,j}_{0,1}y + \dots + a^{i,j}_{t,t}x^ty^t$. For $i \in \{1,\dots,n\}$, user U_i stores the personal key: $S_i = \{N, i, s_{1,1}(i,i), \dots, s_{m,1}(i,i), s_{1,2}(i,i), \dots, s_{m,2}(i,i), \dots\dots, s_{1,m}(i,i), \dots, s_{m,m}(i,i)\}$

2. *(Broadcast) Let $A, R \subseteq \{U_1,\dots,U_n\}$, $|R| \leq t$, denote the active users and revoked users in session j, respectively. The group manager chooses $W = \{w_1, w_2, \dots, w_t\} \subseteq F_q$ such that the indices of the users in R are contained in W, none of the indices of the users in A are contained in W and $N \notin W$. The broadcast in period $j \in \{1,\dots,m\}$, is $\mathcal{B}^1_j \cup \mathcal{B}^2_j$, where:*

$$\begin{aligned}
\mathcal{B}^1_j &= \{p_{j'}(x) + s_{j',j}(N,x)\}_{j'=1,\dots,j-1} \\
&\cup \ \{K_j + s_{j,j}(N,x)\} \\
&\cup \ \{q_{j'}(x) + s_{j',j}(N,x)\}_{j'=j+1,\dots,m} \\
\mathcal{B}^2_j &= \{w_\ell, \{s_{j',j}(w_\ell,x)\}_{j'=1,\dots,m}\}_{\ell=1,\dots,t}
\end{aligned}$$

3. *(Session key and shares recovery in session j) For all $i \in \{1,\dots,n\}$, U_i is able to recover the polynomial $s_{j,j}(x,i)$ using $\{s_{j,j}(w_\ell,x)\}_{\ell=1,\dots,t}$ by evaluating the polynomials at $x = i$ and interpolating based on the points $(i, s_{j,j}(i,i))$ and $\{(w_\ell, s_{j,j}(w_\ell,i))\}_{\ell=1,\dots,t}$. Then U_i recovers K_j by evaluating $s_{j,j}(x,i)$ at $x = N$, and subtracting this value from $(K_j + s_{j,j}(N,x))|_{x=i}$.*

   Additionally, U_i can interpolate to determine $\{s_{j',j}(x,i)\}_{j'=1,\dots,j-1,j+1,\dots,m}$ and thereby recover shares $\{p_{j'}(i)\}_{j'=1,\dots,j-1}$ and $\{q_{j'}(i)\}_{j'=j+1,\dots,m}$ in a similar manner.

Adding users to the group proceeds as in Construction 1. Provided the underlying field is sufficiently large, the group manager adds a new member in session j' by simply giving the user a unique identity, $i \in F_q$, and personal keys corresponding to the current and future sessions $\{s_{j,\ell}(i,i)\}_{j\in\{j',\dots,m\},\ell\in\{j',\dots,m\}}$ (keys corresponding to past sessions are unnecessary).

Theorem 1 *Construction 3 is an unconditionally secure, self-healing session key distribution scheme with t-revocation capability.*

Proof: Recall that our goal is security against coalitions of size at least t. In the following we show this is true in the context of Definition 2.

KEY DISTRIBUTION. A member U_i recovers K_j and shares of the other keys as described in Step 3, and $z_{i,j} = \{p_1(i), \dots, p_{j-1}(i), K_j, q_{j+1}(i), \dots, q_m(i)\}$, appears to be a randomly distributed subset of F_q to an observer who has only either broadcasts or personal keys. In addition, a set of t users, B such that $U_i \notin B$, $i \notin W$, is unable to determine $s_{j',j}(i,i)$ for any $j' \in \{1,\dots,m\}$, because they are only able to recover points on the polynomial, $s_{j',j}(x,i)$ for which $x = i' \in W$ (and $i \notin W$) and the points on the polynomial, $s_{j',j}(i,x)$ for which $x = i'$ and $U_{i'} \in B$. So, given that the degree of each polynomial is t, $s_{j',j}(i,i)$ still appears to be a randomly distributed value in F_q to the users in B. Since no information on $s_{j'j}(.,.)$ for any $j' \in \{1,\ldots,m\}$, is contained in other broadcasts, it follows that $H(s_{j'j}(i,i)|\{S_i\}_{i\in B}, \mathcal{B}_1,\ldots,\mathcal{B}_m) = H(s_{j'j}(i,i))$.

REVOCATION. It suffices to consider what a set of t revoked users, R, learn from the broadcast: $\cup_{U_{i'}\in R} z_{i',j} = \{s_{j'j}(i',x) : U_{i'} \in R, j' = 1,\ldots,m\}$. Hence, for $i = 1,\ldots,n$, the revoked users know at most t points on the polynomials $\{s_{j'j}(x,i)\}$ (and no points if $U_i \notin R$) so each of the points $\{s_{j'j}(N,i)\}$ appears to the revoked users to be randomly distributed in F_q. Because for all $j = 1,\ldots,m$, and all i, the revoked users have no information of $s_{j'j}(N,i)$, it follows that the revoked users have no information on $s_{j'j}(N,x)$, or consequently, on K_j: $H(K_j|\mathcal{B}_j, \{S_{i'}\}_{U_{i'}\in B}) = H(K_j)$.

SELF-HEALING. Recall from Step 3 of the construction that for $j_1 < j < j_2$, a member U_i learns $q_j(i)$ from $\mathcal{B}_{j_1}$ and $p_j(i)$ from $\mathcal{B}_{j_2}$, hence, $K_j = p_j(i)+q_j(i)$ can be reconstructed from both $\mathcal{B}_{j_1}$ and $\mathcal{B}_{j_2}$. Now consider a set of users, B, who are revoked in sessions j and $j_2 > j$ but active in session $j_1 < j$, and a set of users, C, who are revoked in sessions j and $j_1 < j$ but active in session $j_2 > j$. We show that if B and C are disjoint and $|B\cup C| \leq t$, then the colluding users $B \cup C$, are unable to recover K_j from broadcasts, $\mathcal{B}_{j_1}$ and $\mathcal{B}_{j_2}$. In order to recover K_j, $B \cup C$ must recover $q_j(i)$ from $\mathcal{B}_{j_1}$ and $p_j(i)$ from $\mathcal{B}_{j_2}$, for some i. Because the users in C are revoked in session j_1, $B \cup C$ can only recover $\{q_j(i)\}_{U_i\in B}$, and because the users in B are revoked in session j_2, $B\cup C$ can only recover $\{p_j(i')\}_{U_{i'}\in C}$. Hence, because B and C are disjoint and each of size at most t, and both $p_j(x)$ and $q_j(x)$ are of degree t, they cannot recover K_j.

□

The broadcast size in the above construction is $O((mt^2 + tm)\log q)$. Because Construction 3 is both a key distribution scheme with t-revocation capability *and* a self-healing session key distribution scheme, a lower bound on broadcast size follows from lemmas in Appendix D: $|\mathcal{B}| \geq \max\{t^2 \log q, mt \log q\}$. Hence, there seems to be room for improvement in the broadcast size of Construction 3.

5. Reducing Broadcast Size

In this section, we show how to reduce communication overhead from $O((mt^2 + mt)\log q)$ to $O((t^2 + mt)\log q)$, while adding a moderate amount of additional computation at the user's end. The idea behind the reduction is to decrease the size of $\mathcal{B}_j^2$ in Construction 3 by broadcasting a smaller set of polynomials, $\{s_{m,j}(w,x))\}_{w\in W}$, and making public a pseudorandom permutation σ, with which each user can efficiently generate the necessary remaining polynomials, $\{s_{j'j}(w,x))\}_{j'\in\{1,\ldots,m-1\},w\in W}$. The fact that σ's output is pseudorandom is useful, because it ensures that with high probability, the entire collection of polynomials will appear random,and hence, indistinguishable from the collection generated entirely at randomly in Construction 3. We emphasize however, that the choice of pseudorandom σ is enabling but not absolutely necessary. Following the construction, we discuss other approaches.

Because the smaller set of polynomials from which the others are defined can only be specified once the set of revoked users, and hence the set W, is known, we also need to modify the scheme to ensure that the personal keys allocated to users in the set-up phase don't introduce conflicts.

Before stating the construction, we introduce some new notation to make the exposition simpler. For any polynomial in $F_q[x]$, $f(x) = a_0 + a_1x + \ldots + a_tx^t$, and any permutation of F_q, σ, let $\sigma(f(x)) = \sigma(a_0) + \sigma(a_1)x + \ldots + \sigma(a_t)x^t$.

Construction 4 *A variant of Construction 3 in which overhead is reduced.*

1. *(Set-up) Let t be a positive integer, and let N be an element of F_q such that $N \notin \{1,\ldots,n\}$. The group manager chooses the session keys $K_1,\ldots K_m \in F_q$, and the t-degree polynomials $p_1(x),\ldots p_m(x) \in F_q[x]$ all at random. Note that this determines the polynomials, $q_1(x),\ldots,q_m(x)$ as in Construction 1. In addition, for each $r,j \in \{1,\ldots,m\}$, the group manager defines $h_{r,j}(x)$ to be a randomly chosen polynomial of degree $2t$ in $F_q[x]$. For $i = 1,\ldots,m$, U_i stores the personal key $\{N,i,h_{r,j}(i)\}_{r,j=1\ldots,m}$. Finally, for $j = 1,\ldots,m$, the group manager chooses a bivariate polynomial of degree t in each variable, $s_{m,j}(x,y) \in F_q[x,y]$ at random, and a pseudorandom permutation of F_q, σ. The permutation σ is made public.*

2. *(Broadcast in session j) Let $A,R \subseteq \{U_1,\ldots,U_n\}$, $|R| \leq t-1$, denote the set of active members and the set of revoked users, respectively, in session j. The group manager chooses $W \subseteq F_q$ such that $|W| = t$, the indices of the users in R are in W, the indices of users in A are not, and $N \notin W$. Let $W = \{w_1,\ldots,w_t\}$. For $j' = 1,\ldots,m$ the group manager chooses $\{s_{j'j}(x,y)\}_{j'}$ to be bivariate polynomials in $F_q[x,y]$ of degree t in each variable, such that for all $j' = 1,\ldots,m$ and $i = 1,\ldots,t$, $s_{j'j}(w_i,x) = \sigma^{m-j'}(s_{m,j}(w_i,x))$ The broadcast in period $j \in \{1,\ldots,m\}$, is $\mathcal{B}_j^1 \cup \mathcal{B}_j^2$, where:*

$$\begin{aligned}
\mathcal{B}_j^1 &= \{p_{j'}(x) + s_{j'j}(N,x)\}_{j'=1,\ldots,j-1} \\
&\cup \{K_j + s_{j,j}(N,x)\} \\
&\cup \{q'_j(x) + s_{j'j}(N,x)\}_{j'=j+1,\ldots,m} \\
\mathcal{B}_j^2 &= \{h_{j'j}(x) + s_{j'j}(x,x)\}_{j'=1,\ldots,m} \\
&\cup \{w_i, s_{m,j}(w_i,x)\}_{i=1,\ldots,t}
\end{aligned}$$

3. *(Session key and shares recovery in session j) First, U_i recovers $s_{j'j}(i,i)$ for $j' = 1,\ldots,m$ by evaluating $\{h_{j'j}(x) + s_{j'j}(x,x)\}$ at $x = i$ and subtracting $h_{j'j}(i)$. Each user then applies the publicly known pseudorandom permutation σ to recover $\{s_{j'j}(w_1,x),\ldots,s_{j'j}(w_t,x)\}_{,j'\in\{1,\ldots,m-1\}}$, using the fact that $s_{j'j}(w_i,x) = \sigma^{m-j'}(s_{m,j}(w_i,x))$. Recovery of the session keys and the key shares then proceeds as in Construction 3.*

Adding users in Construction 4 is as simple as it is in Construction 3. Provided the underlying field is sufficiently large, the group manager adds a user in session j by giving the users a unique identifier, $i \in F_q$, and the keys $\{h_{r,\ell}(i,i)\}_{r\in 1,\ldots,m,\ell\in\{j,\ldots,m\}}$.

To see that the choice of a pseudorandom permutation facilitates the construction, but is not essential, consider algebraic attacks in which a user U_i who legitimately learns $q_j(i)$ (for example) and then, when revoked in session j_1, uses this knowledge to recover $s_{j,j_1}(N,i)$ and then exploits an algebraic relationship between $s_{j_1,j_1}(x,y)$ and $s_{j,j_1}(x,y)$ to learn session key, K_{j_1}. The algebraic relationship might be as simple as, $s_{j,j_1}(N,i) = s_{j_1,j_1}(N,i)$, then $K_{j_1} = K_{j_1} + s_{j_1,j_1}(N,x)|_{x=i} - s_{j,j_1}(N,i)$. Using a pseudorandom permutation ensures that with high probability the resulting $s_{j'j}(x,y)$ polynomials chosen by the group manager in step 2, will be sufficiently different and the construction will not be vulnerable to such attacks. Although it is possible to accomplish this without a pseudorandom permutation, it is not possible for all permutations. Consider the extreme case of the identity permutation. If σ is the identity permutation, then it is possible for the group manager to choose $s_{j'j}(x,y) = s_{m,j}(x,y)$ for $j', j \in \{1,\ldots,m\}$. The resulting construction is vulnerable to exactly the kind of attack we just described. At the other end of the spectrum, it is also possible to use a truly random permutation to reduce overhead. However, since this potentially places a heavy computational burden on each user (note that in Construction 3 the burdens of unconditional security are only experienced by the group manager), we don't propose such an approach. Hence, we choose to use a pseudorandom permutation in our construction, while noting that there are other secure options.

The proof of security for this construction is in Appendix E. We state the theorem here for completeness.

Theorem 2 *Construction 4 is a self-healing session key distribution scheme with t-revocation capability.*

6. Extending the Lifetime

After a set of m sessions has expired in Constructions 3 and 4, some rekeying of the users is necessary before distributing new session keys. This is so because the state of the system has changed as a result of the broadcasts. For example, in each construction, portions of the personal keys of the revoked users are made public. One solution to this problem is to distribute a new set of secret keys to each user, and proceed as before. Another solution is to use a technique that originated in [16] and is used in [29], which can be described as Shamir secret sharing in the exponent of a generator g, of a cyclic group, G. Moving operations to the exponent allows each user to *evolve* their secret keys from one set of m sessions to the next, thus making the scheme *long-lived*, meaning the scheme can continue without any unicasts from the group manager. This is accomplished through the broadcast of random values at the end of a set of m sessions, by the group manager. Each user (revoked or not) is able to use the random values to calculate their own new personal key. This results in significant bandwidth savings over the naive approach of sending each user a new personal key via unicast, because if each user stores r keys, then r random values must be sent, in contrast to rn unicasts in the naive approach. The savings are reduced by a constant factor, however, because the former approach requires a larger underlying group size (roughly, 160 bits) in order to ensure that the Decision Diffie-Hellman problem is hard.

This technique is applicable to both Constructions 3 and 4. We demonstrate it here for Construction 3 only, because the extension is somewhat simpler and all of the important underlying ideas are illustrated.

The theorem following Construction 5 shows that the construction is secure provided the Decision Diffie-Hellman (DDH) assumption is hard. We informally state the assumption here, referring the reader to [1] for a more precise and detailed discussion and to [29, 10] for examples of proofs of reduction to the DDH problem. DDH is defined for any cyclic group G and generator g. The DDH assumption is that it is difficult to distinguish between the distributions of (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b, and c are chosen randomly in $\{1,\ldots,|G|\}$. DDH is believed to be intractable in groups of large prime order.

Before beginning the construction it is helpful to introduce some additional notation. Given $f(x) = a_0 + a_1x + \ldots + a_tx^t \in G[x]$, let $g^{f(x)} = (g^{a_0},\ldots,g^{a_t})$.

Construction 5 *A Long-lived Variant of Construction 3.*

1. *(Initial Set-up) Let t be a positive integer, g a generator of a subgroup $Z_p \subseteq F_q^*$, of prime order p, and $N \in Z_p$ be such that $N \notin \{1,\ldots,n\}$. The group manager chooses m^2 polynomials in $Z_p[x,y]$ at random, $\{s_{r,j}(x,y)\}_{r,j\in\{1,\ldots,m\}}$, where for each r, j, $s_{r,j}(x,y) = a^{r,j}_{0,0} + a^{r,j}_{1,0}x + a^{r,j}_{0,1}y + \ldots + a^{r,j}_{t,t}x^ty^t$. For $i \in \{1,\ldots,n\}$, user U_i stores the personal key: $S_i = \{N, i, s_{1,1}(i),\ldots,s_{m,1}(i,i), s_{1,2}(i,i),\ldots,$ $s_{m,2}(i,i),\ldots,s_{1,m}(i,i),\ldots,s_{m,m}(i,i)\}$*

2. *(Set-up for the αth set of m sessions) The group manager randomly chooses integers $v^{\alpha}_{1,1}, \ldots, v^{\alpha}_{m,m} \in Z^*_q$ and broadcasts $g^{v^{\alpha}_{1,1}}, \ldots, g^{v^{\alpha}_{m,m}}$. For $i = 1, \ldots, n$, U_i computes a new personal key, $\{g^{v^{\alpha}_{j',j} s_{j',j}(i,i)}\}_{j',j \in \{1,\ldots,m\}}$. The group manager randomly chooses $K^{\alpha}_1, \ldots, K^{\alpha}_m \in Z_p$ and the t-degree polynomials $p^{\alpha}_1, \ldots, p^{\alpha}_m \in Z_p[x]$. Note that this determines the polynomials $q^{\alpha}_1, \ldots, q^{\alpha}_m \in Z_p[x]$ as in Construction 3.*

3. *(Broadcast in session j of the αth set of m sessions) Let $A, R \subseteq \{U_1, \ldots, U_n\}$, $|R| \leq t$, denote the active users and the revoked users, respectively. The group manager chooses $W \subseteq Z_p$ such that $|W| = t$, the indices of the revoked users are contained in W and the indices of the active users are not, and $N \notin W$. The broadcast in period $j \in \{1, \ldots, m\}$, is $\mathcal{B}^1_j \cup \mathcal{B}^2_j$, where:*

$$\begin{aligned}
\mathcal{B}^1_j &= \{g^{p_{j'}(x)+v^{\alpha}_{j',j} s_{j',j}(N,x)}\}_{j'=1,\ldots,j-1} \\
&\cup \{g^{K^{\alpha}_j + v^{\alpha}_{j,j} s_{j,j}(N,x)}\} \\
&\cup \{g^{q_{j'}(x)+v^{\alpha}_{j',j} s_{j',j}(N,x)}\}_{j'=j+1,\ldots,m} \\
\mathcal{B}^2_j &= \{w, g^{v^{\alpha}_{j',j} s_{j',j}(w,x)}\}_{w \in W, j' \in \{1,\ldots,m\}}
\end{aligned}$$

4. *(Session key and shares recovery) U_i recovers $\{g^{v^{\alpha}_{j',j} s_{j',j}(N,i)}\}_{j' \in \{1,\ldots,m\}}$ using $\{g^{v^{\alpha}_{j',j} s_{j',j}(i,i)}\}_{j' \in \{1,\ldots,m\}}$ and $\{g^{v^{\alpha}_{j',j} s_{j',j}(w,i)}\}_{w \in W, j' \in \{1,\ldots,m\}}$. This enables U_i to recover the jth session key $g^{K^{\alpha}_j}$ and the shares, $\{g^{p^{\alpha}_{j'}(i)}\}_{j'=1,\ldots,j-1}$ and $\{g^{q^{\alpha}_{j'}(i)}\}_{j'=1,\ldots,j-1}$.*

Note that $2t + 1$ users can pool their personal keys and reconstruct $\{s_{\ell,j}(x,x)\}_{\ell,j}$, and then these users are able to retrieve session keys for the lifetime of the scheme. Hence, even with this long-lived self-healing scheme, occasionally "re-starting" the scheme by securely sending each user a fresh personal key, is desirable.

The proof of security for this construction is in Appendix E. We state the theorem here for completeness.

Theorem 3 *Construction 5 is a computationally secure, long-lived, self-healing session key distribution scheme with t-revocation capability.*

7. Practical Issues

A number of practical issues need to be addressed before deploying the constructions of this paper in real-world applications. First, we need to consider the scenarios in which self-healing key distribution schemes would be applied and determine the system parameter values that are appropriate for these scenarios. Then, we need to ensure that an efficient implementation of our schemes – with the chosen parameter values – is possible.

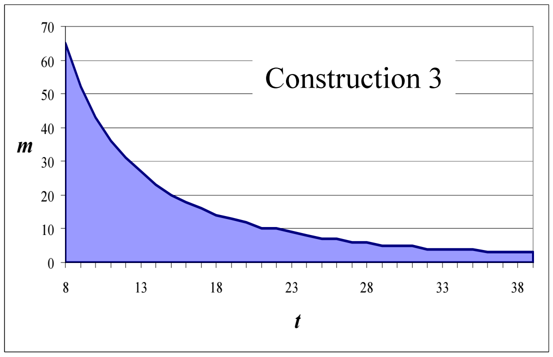
The work described in this paper is part of a larger project investigating secure group communication for large, dynamic groups. In particular, the project is concerned with groups with 10,000 (or more) members, in which membership may change frequently (possibly every few seconds). Our schemes are well-suited for this setting because the system parameters affecting broadcast size are either independent of the number of members (as is the case for m, the number of sessions, and the key size, $\log q$, whose value is determined by the necessary cryptographic strength, which is typically much larger than the group size) or grow much more slowly (as does the collusion resistance, t). The actual session length will vary according to the key size used and the rate of change in group membership. In practice, we anticipate it will be in the range of a few seconds to a minute.

We determined that q (recall, all operations are in the finite field F_q) should be at least 2^{64}, *i.e.*, a 64-bit number. This ensures that we can broadcast session keys $K_1, \ldots, K_m$ that are also 64 bits long. Presumably, these session keys will be used in a symmetric cipher such as AES, for which a 64-bit key currently provides reasonable security for a short-lived session key.

The maximum packet size in an IPv4-based network is 64KB. Figure 3 shows possible values for m and t, given this constraint. We note that larger broadcasts are less likely to reach their destinations: If we assume packets are lost independently at random at a rate of 1%, and consider a key distribution broadcast made out of 45 such packets (fragments)[2], then there is a 36% chance that one fragment, and hence the broadcast as a whole, will not reach its destination. If the loss rate reaches (a fairly high) 5%, then the probability that our 64KB broadcast goes through is only 10%. In other words, recipients will see only every tenth broadcast. Choosing m to be between 10 and 20 should address this problem as users will, in fact, very likely be able to recover missed session keys through self-healing.

Fixing m to be between 10 and 20 leaves us with values for t between 15 and 20 for Construction 3, and even larger values for Construction 4. The dynamic nature of the group supports providing only a moderate degree of collusion resistance. Because the group is dynamic, collusions formed in a previous session may not be as useful in the current one (*e.g.*, if a member is now revoked, and hence, doesn't have useful information on the current session key), so a certain amount of *new* collusion may be necessary in each session. The difficulty in forming useful collusions within a short time period reduces the needed degree of collusion resistance. Therefore, the above mentioned values for t and

[2]Most IP stacks will break large UDP packets down to 1500-byte Ethernet-packet-sized fragments.



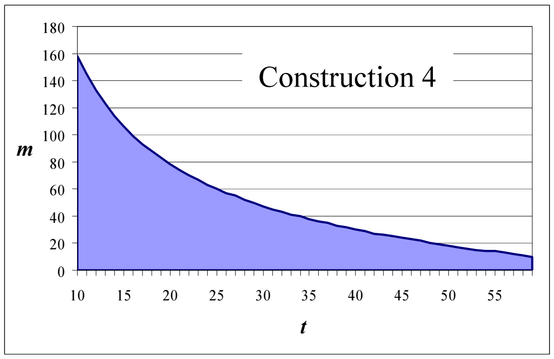


Figure 3. Possible values for m and t, given a maximum broadcast size of 64K.

m should be adequate for most applications.

If the high likelihood of broadcast loss and the associated high latency for key recovery (*i.e.,* it may take a few sessions until we learn the key of a lost broadcast) associated with Construction 3 is unacceptable for a given application, there are two straightforward solutions. First, the application can use Construction 4 and/or use smaller values for t and m. This will decrease the size of the broadcast substantially, and lower the probability of broadcast loss (in which case a smaller m is sufficient). Second, an implementation in which the group manager broadcasts the $m - 1$ shares for previous and future keys, and the current session key, independently, can be used (*i.e.*, the group manager performs the fragmentation). With such an implementation, m smaller broadcasts are used to send the same information as is currently done in one broadcast. Every single one of the smaller broadcasts has a higher probability of reaching its target, and the receivers can still use the subset of shares they receive to self-heal on some of the missed broadcasts.

One concern about the schemes presented here is that they are defined over a fixed period of m sessions, and hence, session keys corresponding to sessions late in the sequence are more vulnerable to packet loss because there is less opportunity to form a "sandwich" of received packets. This may also be true of session keys corresponding to the beginning sessions (although, if unicasts are already being used to distribute personal keys, it might make sense to send the first key distribution packet via unicast, as well). By making m a bit larger, we can ensure that with high probability each user will either receive, or be able to recover via self-healing, most of the session keys. However, there is still the issue of distributing new personal keys to each in member in order to deploy the self-healing key distribution for a new round of m sessions. In Section 6 and Appendix E, we discuss a way to eliminate the need to individually re-key every group member after every m sessions. Furthermore, we are also currently working on *sliding-window* versions of the schemes presented here, in which *any* two packets that "sandwich" a session sufficiently closely, can be used to recover that session's key.

8. Open Problems

We have shown that self-healing key distribution provides reliable multicast session key distribution in a manner that is stateless and conducive to traceability. A reasonable degree of resistance to both adversarial coalitions and network packet loss can be achieved with overhead of just a single UDP packet per session. In addition, members who experience packet loss can recover missed session keys efficiently upon receipt of a single additional packet. Many open questions remain. We have presented constructions in which the number of sessions, m, contributes linearly to the broadcast size. It would be interesting to explore computational versions of self-healing key distribution further, as in this setting it may be possible to remove the m term entirely. In addition, it's unclear whether the degree of t in the broadcast size can be reduced.

In a sliding-window variation of this approach, users are able to recover any lost session key, provided bursts of loss amongst the key distribution packets are of length less than m. When self-healing is implemented over a fixed set of m sessions, there is a reduced resistance to loss for session keys associated with the last few sessions–sliding window self-healing corrects this problem by ensuring that members recover information about session keys in a window of constant size around the current session, no matter what the actual session number is. Many of the techniques of this paper can be used to implement a sliding window self-healing scheme. The fact that new personal keys are needed with every session presents an interesting problem. The techniques of Section 6 provide one solution, but since the cost of the modular exponentiations involved may be prohibitive, it is interesting to look for other alternatives.

9. Acknowledgments

The authors would like to thank Alice Silverberg, Hao-Chi Wong and the anonymous referees for helpful comments on an earlier version of this paper, and Tom Berson and Ralph Merkle for several interesting discussions.

This work was partially sponsored by DARPA grant N66001-00-1-8921.

References

[1] D. Boneh. *The Decision Diffie-Hellman Problem.* In Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science **1423**, pp. 48–63, 1998.

[2] A. Beimel and B. Chor. *Interaction in Key Distribution Schemes.* In Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science **773**, pp. 444–455.

[3] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution.* In Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science **773**, pp. 232–249.

[4] M. Bellare and P. Rogaway. *Provable Secure Session Key Distribution-The Three Party Case.* In 27th ACM Symposium on the Theory of Computing, May 1995.

[5] S. Berkovit. *How to Broadcast a Secret.* In Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science **547**, pp. 536-541.

[6] S. Blake-Wilson and A. Menezes. *Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques.* In the Security Protocols Workshop '97.

[7] R. Blom. *Non-Public Key Distribution.* In Advances in Cryptology - Crypto '82, Plenum Press, pp. 231-236.

[8] C. Blundo, L. Frota Mattos and D. Stinson. *Trade-offs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution.* In Advances in Cryptology - Crypto '96, Lecture Notes in Computer Science **1109**, pp. 387–400.

[9] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. *Perfectly secure key distribution for dynamic conferences.* In Information and Computation, **146** (1), 1998, pp 1-23.

[10] D. Boneh and M. Franklin. *An Efficient Public Key Traitor Tracing Scheme*. In Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science **1666**, pp. 338–353.

[11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas. *Multicast Security: A Taxonomy and Some Efficient Constructions*. In IEEE INFOCOM, **2** (March 1999), pp. 708–716.

[12] R. Canetti, T. Malkin and K. Nissim. *Efficient Communication-Storage Tradeoffs for Multicast Encryption*. In Advances in Cryptology - Eurocrypt '99, Lecture Notes in Computer Science **1592**, pp. 459–474.

[13] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[14] G. Davida, Y. Desmedt and R. Peralta. *A Key Distribution System Based on any One-Way Function.* In Advances in Cryptology - Eurocrypt '89, Lecture Notes in Computer Science **434**, pp. 75-79.

[15] C. Dwork, J. Lotspiech, M. Naor. *Digital Signets: Self-Enforcing Protection of Digital Information.* In the ACM Symposium on Theory of Computing (STOC), 1996.

[16] P. Feldman. *A Practical Scheme for Non-Interactive Secret Sharing.* In Proc. 28th IEEE Symposium on Foundations of Computer Science, 1987, pp. 427–437.

[17] A. Fiat and M. Naor. *Broadcast Encryption.* In Advances in Cryptology - Crypto '93, Lecture Notes in Computer Science **773**, pp. 480–491.

[18] W. Fumy and M. Munzert. *A Modular Approach to key Distribution.* In Advances in Cryptology - Crypto '90, Lecture Notes in Computer Science **537**, pp. 274–283.

[19] G. Hanaoka, T. Nishioka, Y. Zheng and H. Imai. *An Efficient Hierarchichal Identity-Based Key-Sharing Method Resistant Against Collusion-Attacks.* In Advances in Cryptology - Asiacrypt '99, pp. 348-362.

[20] M. Just, E. Kranakis, D. Krizanc and P. van Oorschot. *On Key Distribution via True Broadcasting.* In ACM Conference on Computer and Communications Security, 1994, pp. 81–88.

[21] R. Kumar, S. Rajagopalan and A. Sahai. *Coding constructions for blacklisting problems without computational assumptions.* In Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science **1666**, pp. 609–623.

[22] H. Kurnio, R. Safavi-Naini, W. Susilo and H. Wang. *Key Management for Secure Multicast with Dynamic Controllers.* Fifth Australasian Conference on Information Security and Privacy, ACISP 2000, Lecture Notes in Computer Science **1841**, pp. 178–190.

[23] T. Matsumoto and H. Imai. *On the Key Predistribution System: A Practical Solution to the Key Distribution Problem*. In Advances in Cryptology - Crypto '87, Lecture Notes in Computer Science **293**, pp. 185–193.

[24] T. Matsumoto, Y. Takashima and H. Imai. *On Seeking Smart Public-Key Distribution Ssystems.* In Transactions of the IECE of Japan, February 1986, pp. 99–106.

[25] U. Maurer. *Information-Theoretically Secure Secret-Key Agreement by not Authenticated Public Discussion.* In Advances in Cryptology - Eurocrypt '97, Lecture Notes in Computer Science **1233**, pp. 209–225.

[26] D. McGrew and A. Sherman, *Key establishment in large dynamic groups using one-way function trees.* Submitted.

[27] R. Molva and A. Pannetrat. *Scalable Multicast Security with Dynamic Recipient Groups*. In ACM Transactions on Information and System Security, Vol. 3, No. 3, Aug 2000.

[28] D. Naor, M. Naor and J. Lotspiech. *Revocation and Tracing Schemes for Stateless users.* In Advances in Cryptology - Crypto '01, Lecture Notes in Computer Science **2139**, pp. 41–62.

[29] M. Naor and B. Pinkas. *Efficient Trace and Revoke Schemes*. In Proceedings of Financial Cryptography 2000, Lecture Notes in Computer Science (2001) **1962** pp. 1–20.

[30] B. Pinkas. *Efficient State Updates for Key Management*. In Workshop on Security and Privacy in Digital Rights Management 2001, November 5, 2001.

[31] A. Perrig, D. Song and J. D. Tygar. *ELK, a New Protocol for Efficient Large-Group Key Distribution*. In IEEE Symposium on Security and Privacy (2001), pp. 247–262.

[32] R. Safavi-Naini and H. Wang. *New Constructions of Secure Multicast Re-keying Schemes using Perfect Hash Families*. 7th ACM Conference on Computer and Communication Security, ACM Press, 2000, pp. 228–234.

[33] S. Setia, S. Koussih and S. Jajodia. *Kronos: A Scalable Group Re-Keying Approach for Secure Multicast*. In IEEE Symposium on Security and Privacy (2000).

[34] A. Shamir. *How to Share a Secret*. In Communications of the ACM, **22**, 1979, pp. 612–613.

[35] V. Shoup and A. Rubin. *Session Key Distribution Using Smart Cards.* In Advances in Cryptology - Eurocrypt '96, Lecture Notes in Computer Science **1070**, pp. 321–331.

[36] G. Simmons. *Prepositioned Shared Secret and/or Shared Control Schemes.* In Advances in Cryptology - Eurocrypt '89, Lecture Notes in Computer Science **434**, pp. 436–467.

[37] D. R. Stinson and R. Wei. *Key preassigned traceability schemes for broadcast encryption.* in Selected Areas in Cryptology – SAC '98, Lecture Notes in Computer Science **1556**, pp. 144–156.

[38] P. Syverson and C. Meadows. *Formal Requirements for key Distribution Protocols.* In Advances in Cryptology - Eurocrypt '94, Lecture Notes in Computer Science **950**, pp. 320–331.

[39] D. Wallner, E. Harder and E. Agee. *Key Management for Multicast: Issues and Architectures.* Internet Draft , ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt.

[40] C. Wong, M. Gouda and S. Lam. *Secure Group Communication Using Key Graphs.* SIGCOMM '98. Also, University of Texas at Austin, Computer Science Technical Report TR 97-23.

[41] C. Wong and S. Lam. *Keystone: A Group Key Management Service.* In International Conference on Telecommunications, ICT 2000.

[42] Y. Yacobi. *A Key Distribution Paradox.* In Advances in Cryptology - Crypto '90, Lecture Notes in Computer Science **537**, pp. 268–273.

[43] Y. Yacobi and Z. Shmuely. *On Key Distribution Systems.* In Advances in Cryptology - Crypto '89, Lecture Notes in Comouter Science **435**, pp. 344–355.

A. Information Theory Tools

In this section we give a brief overview of the information theory tools we use. For details on the topics presented here, we refer the reader to [13].

Let X be a random variable that takes values in the the finite set $\mathcal{X}$, according to the probability distribution $\{p(x)\}_{x\in\mathcal{X}}$. The *entropy* of X is defined to be:

$$H(X) = -\Sigma_{x\in\mathcal{X}} p(x) \log p(x)$$

The $\log$ above is to base two, hence entropy can be expressed in bits. Intuitively, entropy is a measure of the amount of information contained in a random variable. For example, the entropy of a variable who's value is determined by a random coin flip is 1; it contains one bit of information as the coin can be heads or tails.

We also makes use of the concepts of conditional entropy and joint entropy. Let Y be a random variable that takes values in the the finite set $\mathcal{Y}$, according to the probability distribution $\{q(y)\}_{y\in\mathcal{Y}}$. The *conditional entropy* of X given Y is:

$$H(X|Y) = \Sigma_{y\in\mathcal{Y}} q(y) H(X|Y=y)$$

To define joint entropy, we need the concept of a joint probability distribution. Let the probability that random variable (X,Y) takes on the value (x,y) where $x\in\mathcal{X}$ and $y\in\mathcal{Y}$ be denoted by $r(x,y)$. The *joint entropy* of X and Y is then:

$$H(X,Y) = -\Sigma_{x\in\mathcal{X}}\Sigma_{y\in\mathcal{Y}} r(x,y)\log r(x,y)$$

The chain rule for entropy state is stated below without proof (see [13]).

Theorem 4 (Chain Rule) *Let $X_1,\ldots,X_t$ be random variables with joint probability distribution denoted by $\{r(x_1,\ldots,x_t)\}$.*
Then $H(X_1,\ldots,X_t) = \Sigma_{i=1}^{t} H(X_i|X_{i-1},\ldots,X_1)$.

We introduce a technical lemma that follows naturally from the properties of the entropy function.

Lemma 5 *Let X, Y, Z denote random variables. If $H(X|Y,W) = 0$ and $H(X|Z,W) = H(X)$, then $H(Y|Z) \geq H(X)$.*

Proof:

$$\begin{aligned}
H(Y|Z) &= H(Y|Z) + H(X|Y,W,Z)\\
&= H(X,Y,W,Z) - H(Z) - H(W|Y,Z)\\
&= H(Z) + H(W|Z) + H(X|Z,W)\\
&\quad + \ H(Y|X,Z,W) - H(Z)\\
&\quad - \ H(W|Y,Z)\\
&= H(X) + H(W|Z) + H(Y|X,Z,W)\\
&\quad - \ H(W|Y,Z)\\
&\geq H(X) + H(Y|X,Z,W)\\
&\geq H(X)
\end{aligned}$$

□

B. Proof of Security for Construction 1

Lemma 1 *Construction 1 is an unconditionally secure, self-healing, session key distribution scheme (with no revocation capability).*

Proof:

Part three of the Construction 1 describes how member U_i recovers K_j from $\mathcal{B}_j$. Because the polynomials $\{h_j(x)\}_{j\in\{1,\ldots,m\}}$ are chosen randomly, no information about
$z_{i,j} = \{p_1(i),\ldots,p_{j-1}(i),K_j,q_{j+1}(i),\ldots,q_m(i)\}$ can be learned from the broadcast without the help of personal keys, and because $\{p_j(x)\}_{j\in\{1,\ldots,m\}}$ and $\{K_j\}_{j\in\{1,\ldots,m\}}$ are chosen randomly, no information on $z_{i,j}$ can be determined from $\{h_j(x)\}_{j\in\{1,\ldots,m\}}$ without any broadcasts.

The construction is self-healing because for $1 \leq j_1 < j < j_2 \leq m$, and $i\in\{1,\ldots,n\}$, K_j can be reconstructed from $\mathcal{B}_{j_1}$, $\mathcal{B}_{j_2}$ and S_i as follows:

$$\begin{aligned}
p_j(i) &= (h_j(x)+p_j(x))|_{x=i} - h_j(i)\\
q_j(i) &= (h_j(x)+q_j(x))|_{x=i} - h_j(i)\\
K_j &= p_j(i)+q_j(i)
\end{aligned}$$

Finally, because there are no revoked users in this scheme, attacks in which colluding users attempt learn session keys they aren't entitled to, aren't relevant.

□

C. Key Distribution and the Proof of Security for Construction 2

In a key distribution scheme, $\mathcal{D}$, the group manager seeks to establish a new key $k_i \in F_q$, with each user U_i over a broadcast channel. We state the definitions important to unconditionally secure key distribution in words below and, in each case, provide the corresponding information theoretic equation for clarity. Following the definition is a proof of security for the key distribution scheme of Construction 2.

Definition 3 *[Key Distribution.] Let $t,i\in\{1,\ldots,n\}$.*

1. *$\mathcal{D}$ is a key distribution scheme if the following are true:*
   - *(a) For any member U_i, k_i is determined by S_i and $\mathcal{B}$ ($H(k_i|\mathcal{B},S_i)=0$).*
   - *(b) For any set $B \subseteq \{U_1,\ldots,U_n\}$ such that $|B| \leq t$, and any user $U_i \notin B$, the users in B are not able to learn anything about k_i ($H(k_i,S_i|\{S_{i'}\}_{U_i'\in B},\mathcal{B}) = H(k_i,S_i)$).*
   - *(c) No information on $\{k_i\}_{i\in\{1,\ldots,n\}}$ is learned from either the broadcast or the personal keys, alone ($H(k_1,\ldots k_n|\mathcal{B}) = H(k_1,\ldots k_n) = H(k_1,\ldots k_n|S_1,\ldots,S_n)$).*
2. *$\mathcal{D}$ has t-revocation capability, if given any set $R \subseteq \{U_1,\ldots,U_n\}$ such that $|R| \leq t$, the group manager can generate a broadcast $\mathcal{B}$, such that for all $U_i \notin R$, U_i can recover k_i ($H(k_i|\mathcal{B},S_i) = 0$), but the revoked users cannot recover any of the keys ($H(k_1,...,k_n|\mathcal{B},\{S_{i'}\}_{U_{i'}\in R}) = H(k_1,...,k_n)$).*

Lemma 4 *Construction 2 is an unconditionally secure key distribution scheme with t-revocation capability.*

Proof: A member U_i recovers the new key k_i as described in step three Step 3 of the construction. To prove resistance to collusion, first consider a set of t colluding users A, and a member $U_i \notin A$. We show that it is impossible for coalition A to learn $f(i)$ because knowledge of $f(i)$ implies knowledge of $s(N,i)$ and the coalition has no information on the latter value. As described in Step 3 of the construction, the users in A can determine $s(x,\ell)$ for every $U_\ell \in A$. In fact, $\cup_{U_\ell \in A} z_{i,j} = \{s(x,\ell) : U_\ell \in A\}$ Hence, A knows the following points on the polynomial $s(N,x)$: $\{s(N,i) : i \in A\}$. Because $s(N,x)$ is a polynomial of degree t and the colluding users only have t points on it, $s(N,i)$ still appears to be randomly distributed to the colluding users, and so, $f(i)$ appears to be randomly distributed to A: $H(f(i)|f(x)+s(N,x), \{(w, s(w,x)) : w \in W\}, \{(\ell, s(\ell,\ell)) : \ell \in A\}) = H(f(i))$.

In addition, note that if A consists entirely of revoked users then the coalition knows only $t-1$ points on each of the polynomials $\{s(x,\ell) : \ell \in A\}$, which implies that $f(\ell)$ also appears randomly distributed to A for every $\ell \in A$. □

D. Lower Bounds

KEY DISTRIBUTION WITHOUT REVOCATION We prove lower bounds on communication and user storage in unconditionally secure, self-healing session key distribution schemes (see Definition 2). The bounds agree with the intuition that a user must have independent pieces of secret information for each session, and that the size of the broadcast messages is correlated with the number of sessions and the collusion resistance. Construction 1 is essentially tight with both of these bounds.

Lemma 2 *In an unconditionally secure session key distribution scheme, if user U_i is entitled to all m session keys, then $H(S_i) \geq m \log q$, for each $i \in \{1,\ldots,n\}$.*

Proof: Since $H(K_1,\ldots,K_m|\mathcal{B}_1,\ldots,\mathcal{B}_m,S_i) = 0$ and $H(K_1,\ldots,K_m|\mathcal{B}_1,\ldots,\mathcal{B}_m) = H(K_1,\ldots,K_m)$, it follows from Lemma 5 that $H(S_i) \geq H(K_1,\ldots,K_m)$. The session keys are chosen independently at random, so, $H(K_1,\ldots,K_m) = H(K_1)+\ldots+H(K_m) = m\log q$, and the result follows. □

The following result relates the size of the broadcasts in each session to the number of sessions and the collusion resistance.

Lemma 3 *In an unconditionally secure, self-healing session key distribution scheme (with no revocation), $H(\mathcal{B}_j)$ is $\Omega(mt)\log q$.*

Proof: First note that $H(\mathcal{B}_j) \geq H(z_{1,j},\ldots,z_{n,j})$ follows from Lemma 5 and the following two equalities:

$$H(z_{1,j},\ldots,z_{n,j}|\mathcal{B}_j,k_1,\ldots,k_n) = 0$$
$$H(z_{1,j},\ldots,z_{n,j}|k_1,\ldots,k_n) = H(z_{1,j},\ldots,z_{n,j})$$

So, it suffices to prove a lower bound on $H(z_{1,j},\ldots,z_{n,j})$.

$$\begin{aligned} H(z_{1,j},\ldots,z_{n,j}) &\geq H(z_{1,j},\ldots,z_{t,j}) \\ &= H(z_{1,j}) + H(z_{2,j}|z_{1,j}) \\ &\quad + \ldots + H(z_{t,j}|z_{t-1,j},\ldots,z_{1,j}) \end{aligned}$$

Applying Lemma 5 once again and using the fact that, due to self-healing, for $1 \leq s \leq t$:

$$H(K_{j+1},\ldots,K_{m-1}|z_{s,j},z_{s,m}) = 0$$
$$H(K_{j+1},\ldots,K_{m-1}|z_{1,j},\ldots,z_{s-1,j},z_{s,m}) = H(K_{j+1},\ldots,K_{m-1})$$

It follows that
$H(z_{s,j}|z_{1,j},\ldots,z_{s-1,j}) \geq (m-1-j)\log q$.

Note that for $1 \leq s \leq t$ the following two equalities also hold (again, by self-healing):

$$H(K_2,\ldots,K_{j-1}|z_{s,j},z_{s,1}) = 0$$
$$H(K_2,\ldots,K_{j-1}|z_{1,j},\ldots,z_{s-1,j},z_{s,1}) = H(K_2,\ldots,K_{j-1})$$

Hence, from Lemma 5 it is also true that $H(z_{s,j}|z_{1,j},\ldots,z_{s-1,j}) \geq (j-2)H(K_j)$. Combining these two lower bounds, it follows that for $1 \leq s \leq t-1$, $H(z_{s,j}|z_{1,j},\ldots,z_{s-1,j}) \geq (\frac{m}{2}-2)H(K_j)$, and so, $H(\mathcal{B}_j) \geq t(\frac{m}{2}-2)\log q$. □

E. Proofs of Security for Constructions in the Computational Setting

In this appendix we prove the security of Constructions 4 and 5.

Theorem 2 *Construction 4 is a self-healing session key distribution scheme with t-revocation capability.*

Proof: Because $\forall\, j', j$, the degree of $h_{j',j}(x)$ is $2t$, it takes the collusion of $2t+1$ users to compromise the scheme through knowledge of those polynomials. Hence, this modification of Construction 3 does not reduce the collusion resistance. Once each user has calculated their personal keys

$\{s_{j'j}(i,i)\}_{j'=\{1,\ldots,m\}}$, broadcast, $\mathcal{B}_j$ of Construction 4, is indistinguishable from broadcast, $\mathcal{B}_j$ of Construction 3, to a polynomial time adversary, because σ is a pseudorandom permutation. Consequently, the properties of self-healing and t-revocation capability are inherited from Construction 3. Hence, it suffices to show that there exist polynomials, $\{s_{j'j}(x,y)\}_{1\le j'\le m}$, that satisfy the constraints of the construction. This follows because for each $j'=1,\ldots,m$, $\mathcal{B}_j^2$ provides $t(t+1)$ equations in the $(t+1)^2$ coefficients of $s_{j'j}(x,y)$. Further, these equations are linearly independent, so $t+1$ of the coefficients may be chosen at random and the remaining coefficients are determined by the equations. For completeness, we list those equations for a particular $j'\le j$, where

$$s_{j'j}(x,y)=c_{0,0}+c_{0,1}x+c_{1,0}y+\ldots+c_{t,t}x^t y^t$$

and

$$\forall w\in W, s_{m,j}(w,x)=b_0^w+b_1^w x+\ldots+b_t^w x^t.$$

$$\begin{aligned}s_{j'j}(w_1,x)&=(c_{0,0}+c_{1,0}w_1+\ldots+c_{t,0}{w_1}^t)+\ldots\\&\quad\ldots+(c_{0,t}+c_{1,t}w_1+\ldots+c_{t,t}w_1^t)x^t\\&=\sigma^{m-j'}(b_0^{w_1})+\sigma^{m-j'}(b_1^{w_1})x+\ldots\\&\quad\ldots+\sigma^{m-j'}(b_t^{w_1})x^t\\&\ \vdots\\s_{j'j}(w_t,x)&=(c_{0,0}+c_{1,0}w_t+\ldots+c_{t,0}w_t^t)+\ldots\\&\quad\ldots+(c_{0,t}+c_{1,t}w_t+\ldots+c_{t,t}w_{t-1}^t)x^t\\&=\sigma^{m-j'}(b_0^{w_t})+\sigma^{m-j'}(b_1^{w_t})x+\ldots\\&\quad\ldots+\sigma^{m-j'}(b_t^{w_t})x^t\end{aligned}$$

□

Before proving the final theorem, we state the definition of computationally session key distribution in this setting for completeness.

Definition 4 *[Session Key Distribution] Let $t,i\in\{1,\ldots,n\}$ and $j\in\{1,\ldots,m\}$.*

1. *$\mathcal{D}$ is a session key distribution scheme if for any member U_i, K_j can be efficiently computed from $\mathcal{B}$ and S_i, although if either the set of m broadcasts or the set of n personal keys are considered separately, it is computationally infeasible to compute K_j (or other useful information) from either set. In addition, it is computationally infeasible for a set of t users, B, to determine a personal key of a user outside of B.*

2. *$\mathcal{D}$ has t-revocation capability if given any set of revoked users $R\subseteq\{U_1,\ldots,U_n\}$ such that $|R|\le t$, the group manager can generate a broadcast $\mathcal{B}_j$, such that for all $U_i\notin R$, K_j can be efficiently computed from $\mathcal{B}_j$ and S_i, but it is infeasible to compute K_j from $\mathcal{B}_j$ and $\{S_{i'}\}_{U_{i'}\in R}$.*

3. *$\mathcal{D}$ is self-healing if the following are true for any $1\le j_1<j<j_2\le m$:*

   (a) *For any U_i who is a member in sessions j_1 and j_2, K_j can be efficiently computed from the set, $\{z_{i,j_1},z_{i,j_2}\}$.*

   (b) *For any disjoint subsets $B,C\subset\{U_1,\ldots,U_n\}$ where $|B\cup C|\le t$, it is infeasible to compute K_j from the set $\{z_{i',j}\}_{U_{i'}\in B,1\le j\le j_1}\cup\{z_{i',j}\}_{U_{i'}\in C,m\ge j\ge j_2}$.*

Theorem 3 *Construction 5 is a computationally secure, long-lived, self-healing session key distribution scheme with t-revocation capability.*

Proof: The self-healing property and t-revocation capability within any set of m sessions, follow from Construction 4. Hence, it suffices to show that to a polynomial time adversary, Construction 5 is as secure as restarting Construction 3 after every m sessions with new secret keys for all users, provided DDH is hard.

For simplicity of exposition we consider the case $m=1$ in detail, and sketch the proof for larger m.

Note that when $m=1$, the scheme reduces to using the key distribution mechanism of Section 3.2 to distribute the session key. We simplify the notation for this case and write the αth broadcast as:

$$\mathcal{B}=\{g^{K_\alpha+v_\alpha(s(N,x))},g^{v_\alpha(s(w_1,x))},\ldots,g^{v_\alpha(s(w_t,x))}\}$$

We show that if t revoked users can determine the session key, then there exists a DDH oracle. To see this, note that a coalition of t users, $U_1,\ldots,U_t$, who are revoked in the αth iteration of the construction, and so aren't entitled to K_α, can be modeled as an algorithm $\mathcal{A}_i$, for some $i\in Z_p$, that takes as input polynomially many (in β) $(2t+1)$-tuples, $(g^{v_\beta},g^{v_\beta s(1,1)},\ldots,g^{v_\beta s(t,t)},g^{v_\beta s(N,1)},\ldots,g^{v_\beta s(N,t)})$, and a challenge $(t+2)$-tuple: $(g^{v_\alpha},g^{v_\alpha s(1,1)},\ldots,g^{v_\alpha s(t,t)},\gamma)$. $\mathcal{A}_i$ is successful if it has a nonnegligible advantage in determining whether $\gamma=g^{v_\alpha(s(N,i))}$ or a random element of Z_p.

Algorithm $\mathcal{A}_i$ can be used to produce an algorithm $\mathcal{A}^{'}$ for solving a variant of DDH in which the problem is to distinguish between the $(2t+3)$-tuples, $(g^a,g^{b_1},\ldots,g^{b_{t+1}},g^{ab_1},\ldots,g^{ab_{t+1}})$ and $(g^a,g^{b_1},\ldots,g^{b_{t+1}},g^{ac_1},\ldots,g^{ac_{t+1}})$, where $a,b_1,\ldots,b_{t+1},c_1,\ldots,c_{t+1}$ are chosen randomly in Z_p. We denote the challenge tuple associated with this

DDH variant by $(g^a, g^{b_1}, \ldots, g^{b_{t+1}}, \gamma_1, \ldots, \gamma_{t+1})$. Because this variant is at least as hard as DDH, showing that algorithm $\mathcal{A}_i$ provides an algorithm for solving the variant suffices to prove the construction is secure (when $m = 1$) assuming DDH is hard. $\mathcal{A}^{'}$ works as follows. $\mathcal{A}^{'}$ first generates polynomially many random v_βs and random values to correspond to $s(1,1), ...s(t,t)$. Using the portion of the challenge that represents $g^{s(N,x)}$, $(g^{b_1}, \ldots, g^{b_{t+1}})$, $\mathcal{A}^{'}$ can determine $g^{s(N,1)}, \ldots, g^{s(N,t)}$, and from $\gamma_1, \ldots, \gamma_{t+1}$, $\mathcal{A}^{'}$ can determine a candidate for $g^{a(s(N,i))}$ which we denote by y. $\mathcal{A}^{'}$ inputs to $\mathcal{A}_i$ the tuples $(g^{v_\beta}, g^{v_\beta s(1,1)}, \ldots, g^{v_\beta s(t,t)}, g^{v_\beta s(N,1)}, \ldots, g^{v_\beta s(N,t)})$ for all β, and the challenge, $(g^a, g^{b_1}, \ldots, g^{b_{t+1}}, y)$ and outputs the answer provided by $\mathcal{A}_i$. Note that $s(x,y)$ is not over-constrained because $N \notin \{1, \ldots, n\}$, and so the inputs to $\mathcal{A}_i$ are consistent. To see what advantage this gives $\mathcal{A}^{'}$ recall that $\mathcal{A}_i$ returns "yes" if it believes $y = g^{a(s(N,i))}$. So, a positive answer from $\mathcal{A}_i$ only shows that $g^{f(x)} = (g^{b_1}, \ldots, g^{b_{t+1}})$ agrees with $g^{a(s(N,x))}$ at $x = i$. This implies that $f(i) \equiv a(s(N,i)) mod p$. There are p^t such polynomials, $f(x) \in Z_p[x]$, of degree t and only one of them is $a(s(N,x))$, hence the probability that a randomly chosen polynomial, that's different from $s(N,x)$, agrees with $s(N,x)$ at $x = i$, is $\frac{p^t - 1}{p^{t+1}} < \frac{1}{p}$. This implies that the advantage of $\mathcal{A}^{'}$ is at least $(1 - \frac{1}{p})$ times the advantage of $\mathcal{A}$.

To extend the proof to $m > 1$, note that because each of the polynomials in the set $\{s_{r,j}(x,y)\}_{r,j \in \{1,\ldots,m\}}$ are chosen independently at random, the proof technique can essentially be repeated m times to show that t revoked users are unable to determine anything about $g^{s_{j',j}(N,x)}$ for any j', j, if DDH is hard. $\square$