Concrete Analysis and Trade-Offs for the (Complete Tree) Layered Subset Difference Broadcast Encryption Scheme

Sanjay Bhattacherjee and Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute
203, B.T.Road, Kolkata, India - 700108.
sanjay.bhattacherjee@gmail.com and palash@isical.ac.in

Abstract

Two key parameters of broadcast encryption (BE) schemes are the transmission size and the user storage. Naor-Naor-Lotspiech (2001) introduced the subset difference (SD) scheme achieving a good trade-off between these two parameters. Halevy-Shamir (2002) introduced the idea of layering to reduce user storage of the NNL scheme at the cost of increased transmission overhead. Here, we introduce several simple ideas to obtain new layering strategies with different trade-offs between user storage and transmission overhead. We define the notion of storage minimal layering and describe a dynamic programming algorithm to compute layering schemes for which the user storage is the minimum attainable using layerings. Further, the constrained minimization problem is considered. A method is described which yields BE schemes whose transmission overhead is not much more than the SD scheme but, whose user storage is still significantly lower.

Finally, an $O(r \log^2 n)$ algorithm is obtained to compute the average transmission overhead for any layering based scheme where r out of n users are revoked. This algorithm works for any layering strategy and also for arbitrary number of users. The algorithm has been used here to generate all data for the average transmission overhead.

Keywords: Broadcast encryption; subset difference; layering; transmission overhead; user storage

1 Introduction

Digital rights management systems like Pay-TV and content protection in HD-DVD and Blu-Ray discs can be modelled as follows. There is a set of users and a centre which broadcasts messages. For each message, the centre decides on a set of *privileged* users which should be able to access the message while the other users (*revoked*) should not be able to do so. A cryptographic system achieving such a functionality is called a broadcast encryption scheme [1, 2].

In a BE scheme, the key pre-distribution centre provides secret information to the users during a set-up phase. A user can derive its keys from this secret information. Each such key corresponds to a subset of users. During transmission in a symmetric key based BE system, a session key is generated and the message is encrypted using the session key. Next the session key is encrypted using several user keys which are determined by the set of privileged users. The additional encryptions of the session key constitute the *header* while the actual encryption of the message is called the *body*. To decrypt, a privileged user can use its secret information to obtain one of the keys with which the session key has been encrypted. Decrypting the appropriate component of the header with this key yields the session key and then decrypting the body with the session key yields the message. The two important parameters of a BE scheme are the length of the header (as given by the number of encryptions of the session key) and the size of secret information to be stored by a user. It is desirable to decrease both as far as possible, but, in most schemes it turns out that decreasing one increases the other.

Naor, Naor and Lotspiech (NNL) [3] introduced an important BE scheme called the subset difference (SD) method. This scheme has been adopted as a standard for content protection in HD-DVD and Blu-ray discs [4]. The NNL-SD scheme is defined for n users where n is a power of two, i.e., $n = 2^{\ell_0}$ for some $\ell_0 \geq 0$. The users are considered to be the leaves of a full binary tree having ℓ_0 levels. Each user needs to store $\ell_0(\ell_0 + 1)/2$ k-bit strings where k is the key length of the underlying symmetric key cryptosystem. If r users are revoked, then the worst case header length (i.e., the number of encryptions of the session key) is $2r - 1$ [3], while the average case header length turns out to be at most $1.25r$ for practical situations (see [5] for a detailed analysis). The header length and the user storage for the SD scheme have been discussed in details in Section 2. The trade-off between user storage and average header length turns out to be very well suited for real-life applications. Further, the scheme itself is quite elegant and reasonably easy to implement.

A later work by Halevy and Shamir [6] introduced a variant of the SD method called the layered subset difference (LSD) scheme. The basic idea is to partition the tree into several layers which gives the name of the scheme. A different trade-off is obtained. User storage is reduced in the LSD method to $\ell_0^{3/2}$ but, the maximum possible header length grows to $4r-3$. In [6], based on simulation results, it is remarked that the average header length is around $2r$. Compared to the SD method, the LSD method reduces the user storage at the cost of increasing the header length.

1.1 Our Contributions

We work within the ambit of the NNL-SD scheme [3] and the idea of layering introduced in [6]. The Halevy-Shamir (HS) layering works for $n = 2^{\ell_0}$ users where ℓ_0 is a perfect square. This limits its usage to very specific number of users $(2^4, 2^9, 2^{16}, 2^{25})$. Two natural extensions of the HS layering strategy that work for values of ℓ_0 that may not be a perfect square (and hence subsume the HS layering strategy) are considered. While both have the same storage requirement, one of them is experimentally seen to have lower average header length. We call this the extended HS or e-HS layering.

The first problem that we tackle is whether the user storage can be lowered further than the e-HS layering strategy. To this end, we introduce the notion of storage minimal layering. For such a strategy, the user storage requirement is the minimum possible that can be obtained from 2-way splitting of SD subsets using layerings. An $O(\ell_0^3)$ time and $O(\ell_0^2)$ space dynamic programming algorithm is presented to compute storage minimal layerings. In the HS layering strategy, the root node of the user tree is treated as a special level. We show that removing this condition yields a scheme where the user storage is significantly reduced while the effect on the average header length is negligible. The resulting storage minimal schemes result in user storages which are between 18% to 24% lower than that required by the (extended) Halevy-Shamir layering scheme. We note that our work does not provide any asymptotic improvement in user storage compared to the Halevy-Shamir scheme. Rather, our work provides concrete improvement in user storage for all practical values of n and also an algorithm to compute the corresponding layering strategies.

Simply minimizing user storage is only one aspect of the problem. We consider the constrained minimization problem whereby one tries to minimize the user storage but, without increasing the actual values of the average header length significantly beyond that achieved by the SD scheme. This is a difficult problem to solve analytically. Instead, we show how to tackle the problem empirically. Given some idea about the number of users that would be revoked, we show how one may use this information to design a layering strategy for which the average header length is almost as small as the SD scheme. The user storage for such a layering scheme is significantly less than that of the SD scheme. Concrete practical examples are provided and it is shown how to tackle this problem for any practical value of the number of users.

We describe an algorithm to compute the expected header length of the layering based SD schemes. This algorithm works for all possible values of the number of users (and not only those values which are powers of two). Assuming that r out of n users are revoked uniformly at random, our algorithm computes the expected header length in $O(r \log^2 n)$ time and $O(\log n)$ space. A simulation based approach can also be used to estimate the average header length. In this approach, for a fixed n and r, a set of r users are randomly revoked and the cover generation algorithm is applied to compute the corresponding header length. This process is repeated many times and the average of the different header lengths is taken to be an estimate of the actual value of the expected header length. Each run will require $O(n)$ space (and hence also $O(n)$ time) to compute the cover and hence the header length. In contrast, our algorithm does away with the need of performing such a simulation study. Given n and r, it directly computes the expected header length when r out of n users are uniformly revoked. Since r will be much smaller than n for practical scenarios, our algorithm will be faster and require much lesser space. The algorithm is of interest in its own right as it will be a useful tool to practitioners who may wish to quickly calculate the average header length for different broadcast scenarios.

1.2 Previous and Related Works

Before [3] and [6], BE schemes using resilient functions and their analysis were proposed in [7, 8]. Subsequent to [3] and [6], there have been some follow-up work analyzing the average header lengths of the SD and LSD schemes. In [9], a generating function is obtained for counting the number of ways p users out of n can be given access privilege so that the header length will be h. For a given n and p, the generating function was used to obtain equations to compute the expected header length. The authors however mentioned that their equations were "complex to compute and difficult to gain insight from". Consequently, they went forward to find *approximations* for the same. In [10], this analysis of the expected header length was continued and it was shown that the standard deviations are small compared to the expected values, as the number of users gets large. Combinatorial analysis of the worst case header length of the SD method has been done in [11]. Lower bounds on the header length of the SD and LSD schemes were found in [12]. All of these works considered the number of users to be a power of two. In [5], this condition was relaxed and the SD method was extended to the CTSD method. A detailed combinatorial and probabilistic analysis of the CTSD method was carried out.

Several works [13, 14] on the combinatorics behind broadcast encryption schemes and different generic bounds on the efficiency parameters have been done. In [15], a generic method for constructing BE schemes from pseudo-random generators was proposed. While NNL [3] and most follow-up works consider BE for stateless user devices, BE schemes for low-state devices were proposed in [16].

Several other BE schemes have been proposed. Linear algebraic techniques have been used in [17] to find a family of broadcast encryption schemes called linear broadcast encryption schemes. The same authors had also proposed key pre-distribution methods based on linear algebraic techniques in [18]. Another interesting work on BE is [19], that works on the idea of "one key per punctured interval". In [19], the worst case header length has been brought down to r or below for the first time, but at the cost of increasing user storage. For $n = 2^{28}$ and $r = 2^{10}$, the header length is below r at the cost of 3.4×10^8 times the storage of the SD scheme. Moreover, the method is more complicated than the SD scheme.

BE schemes have also been proposed in the public key setting. In a public-key based BE, anybody can broadcast to a group of users in the system. We do not consider public key BE in this work and instead refer the reader to relevant work such as [20, 21]. For this paper, by BE we will mean symmetric key BE.

2 Subset Cover Framework

Suppose there are n users. In the subset cover revocation framework, a collection $\mathcal{S}$ of subsets of $\{1, \ldots, n\}$ is defined in a manner such that any set $S \in \mathcal{S}$ has an associated key and any subset of $\{1, \ldots, n\}$ which is not in $\mathcal{S}$ does not have any key associated with it. For a user u, let $\mathcal{S}_u = \{S \in \mathcal{S} : u \in S\}$. User u is given secret information I_u such that it can construct the key associated with any set in $\mathcal{S}_u$.

During the actual broadcast, some users are revoked and some are privileged. Suppose that a subset T of the users are privileged. A cover finding algorithm determines a collection of pairwise disjoint subsets of $\mathcal{S}$ whose union is T. This collection of subsets is called the subset cover. The actual message is encrypted with a session

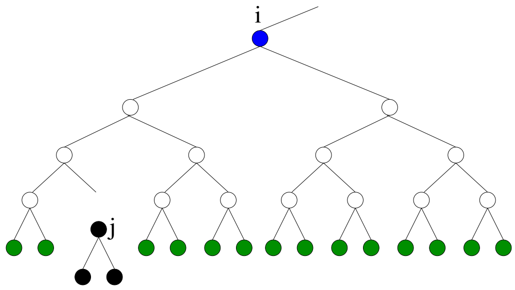

Figure 1: The subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ of users.

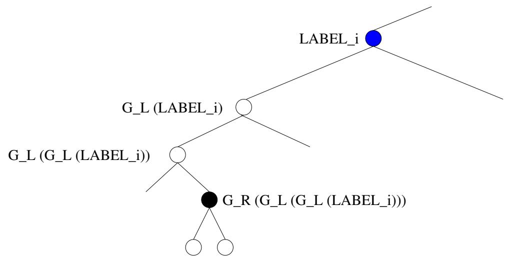

Figure 2: Key for $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is $L_{i,j} = G_M(G_R(G_L(G_L(LABEL_i))))$.

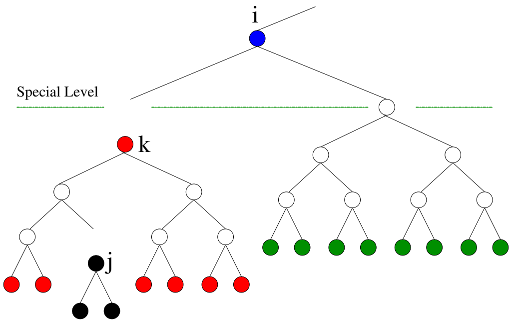

Figure 3: The subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ split into $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}$ and $\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}$.

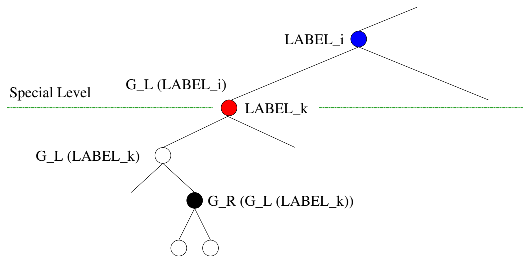

Figure 4: Key for $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}$ is $L_{i,k} = G_M(G_L(LABEL_i))$ and for $\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}$ is $L_{k,j} = G_M(G_R(G_L(LABEL_k)))$.

key and the session key is encrypted with the keys associated with the subsets in the cover. The encrypted message forms the body while the different encryptions of the session key forms the header. So, the number of subsets in the cover determine the header length of the broadcast. Loosely speaking, this number itself is called the header length of the transmission.

To decrypt, a user first determines to which subset of the cover it belongs. Then, using its secret information, it generates the key associated with this subset. Decrypting the appropriate component of the header with this key, the user obtains the session key and then decrypting with the session key the user obtains the actual message.

Two parameters are of crucial interest. The size of the secret information I_u that is to be stored by a user u and the average or expected length of a broadcast header. Basic intuition tells us that as the number of elements in $\mathcal{S}$ grows, it should be possible to cover a privileged set T with lesser number of elements and so the average header length will decrease. On the other hand, as $\mathcal{S}$ grows, the size of $\mathcal{S}_u$ also grows and this should lead to an increase in the size of I_u. Thus, the average header length and the user storage are two competing parameters.

2.1 The Subset Difference Scheme

The SD scheme introduces a major novelty in defining $\mathcal{S}$ such that there is a compact way of representing I_u. In the original SD scheme, the number of users n is a power of 2, say $n = 2^{\ell_0}$. Consider the users to be the leaves of a full binary tree. Each node in the tree represents the users at the leaf level of the tree rooted at that node. Suppose i is a node of the tree and let $\mathcal{S}^{(i)}$ denote the leaves of the subtree rooted at i. Let j be a node in the subtree rooted at i. Then for the SD scheme, the set $\mathcal{S}$ consists of the subsets $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ for all possible choices of node i and all possible nodes $j \neq i$ in the subtree rooted at i as shown in Figure 1. These subsets are called SD subsets.

A clever algorithm is used to define the key associated with an SD subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$. First each node i in the tree is assigned an independent and uniform random label $LABEL_i$. A cryptographically strong pseudo-random generator (PRG) $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$ is used. Let $G(seed)$ be written as the concatenation of 3 k-bit strings $G_L(seed)$, $G_M(seed)$ and $G_R(seed)$. Suppose that a node j in the subtree rooted at node i is reached from node i by the moves 'left', 'left' and 'right'. Then the label of j derived from $LABEL_i$ is $LABEL_{i,j} = G_R(G_L(G_L(LABEL_i)))$ and the key associated with the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is $L_{i,j} = G_M(G_R(G_L(G_L(LABEL_i))))$ as shown in Figure 2. This easily extends to any appropriate pair of nodes i and j. The string $L_{i,j}$ is a k-bit string and the value of k is determined by the key size of the underlying encryption algorithm.

Recall that users are at the leaf level of the tree. The leaf level is numbered 0 and level numbers increase up to ℓ_0 which is the level number of the root. For any user u, the user storage I_u is defined in the following manner. Consider the path from the node u to the root and let i be a node on this path at level $\ell > 0$ of the tree. Let $i_1, \ldots, i_\ell$ be the siblings of the nodes on the path from u to i (including u but not including i). Then for each such i, user u gets the labels $LABEL_{i,i_1}, LABEL_{i,i_2}, \ldots, LABEL_{i,i_\ell}$. The value of ℓ varies from 0 to ℓ_0 and so each user gets $\ell_0(\ell_0+1)/2$ labels. The total size of I_u is $k\ell_0(\ell_0+1)/2$ bits where k is the size of the seed of the PRG. Since k is fixed, it is enough to consider only the number of labels as determining the size of user storage.

The labels provided to a user are sufficient for the user to construct the key corresponding to any element in $\mathcal{S}_u$. To see this suppose that i is a node on the path from u to the root and j is a node in the subtree rooted at i such that $u \in \mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$. Since u is not in $\mathcal{S}^{(j)}$ and both u and j are in the subtree rooted at i, the paths to root from these two nodes intersect for the first time at some node v which is also in the subtree rooted at i. Let v_1 be the first node in the path from v to j. Then v_1 is the sibling of some node v_2 in the path from u to i and so u has $LABEL_{i,v_1}$. From this label, u can generate $LABEL_{i,j}$ by applying G_L and G_R appropriately and so can generate $L_{i,j} = G_M(LABEL_{i,j})$. This $L_{i,j}$ is the key corresponding to the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$. So, u can generate keys for any subset in $\mathcal{S}_u$.

It is also required to argue that u cannot generate keys for any other subset in $\mathcal{S}$. In the SD scheme, any subset in $\mathcal{S}$ is of the form $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$. If u is not in such a subset, then u is either not in $\mathcal{S}^{(i)}$ or it is in $\mathcal{S}^{(j)}$. In either case, it is not too difficult to see that u does not obtain information which allows it to generate $L_{i,j}$. See [3] for more details.

2.2 The Layered Subset Difference Scheme

The point of the LSD scheme is to reduce the user storage in the SD scheme at the cost of increasing the header length. Reduction in the user storage is achieved by reducing the size of $\mathcal{S}$. As in the SD scheme, the LSD scheme also considers the number of users to be of the form 2^{ℓ_0} where the users form the leaves of a full binary tree. The major difference between the SD and the LSD schemes is that in the LSD scheme the levels of the tree are partitioned into layers. Some of the levels are marked as *"special"*. The collection of levels between (and including) two consecutive special levels is called a layer. The levels are numbered with the bottom-most level having the number 0, increasing to the top. The *length* of a layer is the difference between the numbers of the special levels enclosing the layer.

2.2.1 The Halevy-Shamir Layering Strategy

The layering strategy described in [6] is as follows:

> "The root is considered to be at a special level, and in addition we consider every level of depth $k \cdot \sqrt{\log(n)}$ for $k = 1 \ldots \log(n)$ as special (wlog, we assume that these numbers are integers)."

We call this the Halevy-Shamir (HS) layering strategy. It assumes $\sqrt{\ell_0}(= \sqrt{\log n})$ to be an integer and hence ℓ_0 to be a perfect square. The "wlog" in the above statement is valid when one is interested in asymptotic analysis. For concrete values of n, the paper does not describe how to choose a layering scheme. This restricts the use of the scheme to very limited values of n (of the form 2^{ℓ_0} where $\ell_0 = 4, 9, 16, 25$). On the other hand, the authors of [6] consider the case of $n = 2^{28}$ users and suggest a layering strategy with layers of size $6, 6, 6, 5$ and 5. However, they do not give any general description of how to choose the layer lengths when ℓ_0 is not a perfect square. We take up this issue later.

As a consequence of layering, an SD subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is defined to be in $\mathcal{S}$ if either of the following two conditions hold:

- node i is at a special level;
- or, node i is not at a special level but, node j is in the same layer as level i.

This reduces the size of $\mathcal{S}$ and consequently of $\mathcal{S}_u$. As a result, the size of I_u also reduces as we explain below. The distribution of labels is done as follows. Suppose that u is a user (i.e., a leaf node) and i is a node at level ℓ in the path from u to the root and $i_1, \ldots, i_\ell$ are the siblings of the nodes in the path from u to i. If ℓ is a special level, then u is given $LABEL_{i,i_1}, \ldots, LABEL_{i,i_\ell}$ as in the SD scheme. Suppose ℓ is not a special level. Let ℓ' be the first special level below i and consider the segment of the path from u to i which lies between ℓ' and ℓ. Suppose $i_m, \ldots, i_\ell$ are the siblings of the nodes on this segment. Then u gets $LABEL_{i,i_m}, \ldots, LABEL_{i,i_\ell}$. The net effect is that if i is not at a special level, it generates labels only up to the next special level (and not up to the bottom-most level). This leads to the reduction in the user storage.

The reduction in user storage is achieved at the cost of an increase in the header length. Suppose i is not at a special level and j is in the sub-tree rooted at i but not in the same layer as i. The SD scheme would associate the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ to such an (i, j) pair. In the LSD scheme, this set is not present. Instead, the header computation algorithm will cover this set in the following manner. Let k be the node in the first special level as one moves down the path from i to j. The sets $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}$ and $\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}$ are both present in the LSD scheme and it is easy to see that

$$\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)} = \left(\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}\right) \bigcup \left(\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}\right).$$

This can be viewed as a two-way split of the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$. Figure 3 shows the splitting of the subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ of Figure 1. The key assignment to the subsets in Figure 3 is shown in Figure 4. The work [6] also consider the possibility of multi-way split. But, the authors conclude that this leads to further reduction in user storage only for impractical values of the number of users. In this paper, we will not consider multi-way split.

As mentioned earlier, [6] does not mention how to generate a layering strategy when ℓ_0 is not a perfect square. Later in Sections 3(3.1) and 3(3.2) below we look at two natural extensions of the HS layering strategy that can be adopted for 2^{ℓ_0} users for values of ℓ_0 which may not be perfect squares.

3 General Layering Strategy

In general, a layering strategy $\boldsymbol{\ell}$ is denoted by the numbers of the special levels $\ell_0 > \ell_1 > ... > \ell_{e-1} > \ell_e = 0$. Let $\boldsymbol{\ell} = (\ell_0, \ldots, \ell_e)$. The layering strategy has $(e + 1)$ special levels. It is sometimes more convenient to use another formulation to denote the layering. For $1 \le i \le e$, define $d_i = \ell_{i-1} - \ell_i$ so that d_i's are positive integers whose sum is ℓ_0. Conversely, given any sequence of positive integers $\mathbf{d} = (d_1, \ldots, d_e)$ whose sum is ℓ_0, it is possible to define a layering scheme where $\ell_i = \ell_0 - \sum_{j=1}^{i} d_j$.

The user storage for any such layering strategy $\boldsymbol{\ell}$ in general can be calculated as follows. Corresponding to each special level ℓ_i, a user has to store ℓ_i labels. Now consider the nodes in the layer bordered by ℓ_i and ℓ_{i+1}. Corresponding to any non-special level j in this layer a user has to store $j - \ell_{i+1}$ labels. So, the total number of labels that is required to be stored by a user considering both special and non-special levels is given by the following formula.

$$\begin{aligned}
\mathsf{storage}_0(\boldsymbol{\ell}) &= \sum_{i=0}^{e-1} \ell_i + \sum_{i=0}^{e-1} \sum_{j=\ell_{i+1}+1}^{\ell_i-1} (j - \ell_{i+1}) \\
&= \sum_{i=0}^{e-1} \ell_i + \sum_{i=0}^{e-1} \sum_{j=1}^{\ell_i-\ell_{i+1}-1} j \\
&= \sum_{i=0}^{e-1} \ell_i + \frac{1}{2}\sum_{i=0}^{e-1} (\ell_i - \ell_{i+1})(\ell_i - \ell_{i+1} - 1). \qquad (1)
\end{aligned}$$

A recursive description can be obtained as follows.

$$\begin{aligned}
&\mathsf{storage}_0(\ell_0, \ell_1, \ldots, \ell_e) \\
&= \ell_0 + \ell_1 + \cdots + \ell_e + \frac{(\ell_0 - \ell_1)(\ell_0 - \ell_1 - 1)}{2} \\
&\quad + \frac{(\ell_1 - \ell_2)(\ell_1 - \ell_2 - 1)}{2} \\
&\quad + \cdots + \frac{(\ell_{e-1} - \ell_e)(\ell_{e-1} - \ell_e - 1)}{2} \\
&= \ell_0 + \frac{(\ell_0 - \ell_1)(\ell_0 - \ell_1 - 1)}{2} + \mathsf{storage}_0(\ell_1, \ldots, \ell_e). \qquad (2)
\end{aligned}$$

Equation (1) can be formulated in terms of the layer lengths $\mathbf{d} = (d_1, \ldots, d_e)$ as follows.

$$\begin{aligned}
\mathsf{storage}_0(\boldsymbol{\ell}) &= \ell_0(e+1) - \sum_{i=1}^{e} (e - i + 1) d_i \\
&\quad + \frac{1}{2}\sum_{i=1}^{e} d_i(d_i - 1). \qquad (3)
\end{aligned}$$

If all the d_i's are equal to d and $\ell_0 = e \times d$, then $\mathsf{storage}_0(\boldsymbol{\ell})$ is given by $\ell_0(e+d)/2$. This shows that the user storage using e layers of length d each is the same as the user storage using d layers of length e each. If all the layer lengths are equal, then the problem of minimizing the user storage is that of minimizing the sum $e + d$ subject to the constraint $ed = \ell_0$. From this it is easy to see that the minimum value is attained for $e = d = \sqrt{\ell_0}$ and the corresponding value of user storage is $\ell_0^{3/2}$. This justifies the choice made in [6]. Note that the minimization here is in the context of all the layer lengths being equal.

It is easy to note that the layering strategy with each $d_i = 1$ or with $e = 1$ results in the SD scheme. The supplementary material provides some further combinatorial results on general layering strategies.

3.1 The HS Layering with residual bottom layer

Let ℓ_0 be any positive integer and $d \leq \ell_0$. We write $\ell_0 = d(e-1) + p$ where $1 \leq p \leq d$. Then the special levels are

$$\ell_0,\ \ell_0 - d,\ \ell_0 - 2d,\ \ldots,\ \ell - d(e-1),\ 0.$$

So, the tree will have a total of $e+1$ special levels (including the root level ℓ_0 and the leaf level 0) and e layers out of which $e-1$ layers are of length d each and the last layer is of length p. Note that the length p of the bottom-most layer can equal d which will lead to e layers each of length d. We find it convenient to always have level 0 (leaf level) as a special level as this does not have any effect on either the user storage or the header length. The Halevy-Shamir (HS) layering strategy is a special case where ℓ_0 is a perfect square with $d = \sqrt{\ell_0}$ and layer lengths $d, d, \ldots, d, p = d$.

3.2 The e-HS Layering Strategy

We now consider a layering strategy where the layer lengths are balanced. Write $\ell_0 = d(e-1)+p = (e-d+p)d+(d-p)(d-1)$ and define $\mathbf{d}' = (\underbrace{d,\ldots,d}_{e-d+p}, \underbrace{d-1,\ldots,d-1}_{d-p})$. Let $\boldsymbol{\ell}$ be the layering strategy with a residual bottom layer and $\boldsymbol{\ell}'$ be the balanced layering strategy. Then, one can show that $\mathsf{storage}_0(\boldsymbol{\ell}) = \mathsf{storage}_0(\boldsymbol{\ell}')$. (The proof is given in the supplementary material.) So there is no difference between these two strategies in terms of user storage. Experimental results show that the average header lengths for both strategies are similar with that corresponding to the balanced strategy being slightly smaller. As an example, for $\ell_0 = 18$, $\mathbf{d}' = (5,5,4,4)$ yields lesser expected header lengths than $\mathbf{d} = (5,5,5,3)$ for all r between 256 and 16384 while the user storage 75 is the same for both. We call the balanced strategy to be the *extended HS* or *e-HS* layering strategy. This strategy coincides with the layering scheme given in [6] for $n = 28$.

Using (3), it can be verified that storage requirement is $O(\log^{3/2} n)$ for both the e-HS and the residual bottom layer strategies.

3.3 Root at a non-special level

In the HS layering [6] as well as its extensions given in Section 3(3.1) and Section 3(3.2), the root level ℓ_0 is always taken as a special level. It is possible to obtain further reduction in user storage if we allow the root level to be a non-special level. Having the root as a special level contributes ℓ_0 labels to the user storage. If instead the root level is made non-special, then its contribution to the user storage will be $\ell_0 - \ell_1$ labels. Given a sequence of level numbers $\boldsymbol{\ell}$, let $\mathsf{storage}_1(\boldsymbol{\ell})$ be the number of labels required to be stored when the root (top-most) level is not special (and so, ℓ_1 is the first special level). Then the following relation holds.

$$\mathsf{storage}_1(\boldsymbol{\ell}) \quad = \quad \mathsf{storage}_0(\boldsymbol{\ell}) - \ell_1. \tag{4}$$

Combining this with (2) we get the following relation.

$$\begin{aligned}\mathsf{storage}_1(\ell_0,\ldots,\ell_e) \quad = \quad & \frac{(\ell_0-\ell_1)(\ell_0-\ell_1+1)}{2} \\ & +\mathsf{storage}_0(\ell_1,\ldots,\ell_e).\end{aligned} \tag{5}$$

So, not having the root at a special level reduces the storage requirement by ℓ_1 labels. This can be quite significant. Consider the e-HS layering strategy where $\ell_0 = d \times e$ and so $\boldsymbol{\ell} = (\ell_0, \ell_1, \ldots, \ell_e)$ where $\ell_i - \ell_{i+1} = d$ for $0 \le i < e$. In this case, $\mathsf{storage}_0(\boldsymbol{\ell}) = \ell_0^{3/2}$ and $\mathsf{storage}_1(\boldsymbol{\ell}) = \ell_0^{3/2} - (\ell_0 - \ell_0^{1/2})$.

It is important to understand the effect on the header length when the root level is not special. During the computation of the cover, suppose that the root generates an SD subset, i.e., the SD cover finding algorithm returns a subset of the form $\mathcal{S}^{(0)} \setminus \mathcal{S}^{(j)}$. Since the root is not at a special level, this subset may be split into two if j is not in the first layer. We argue that for reasonable values of r (the number of revoked users), this effect is negligible. In fact, the argument is that the probability of the root generating an SD subset itself is small.

The root generates an SD subset only if exactly one of the two subtrees of the root node contains all the revoked users. Intuitively this probability is low even for moderate values of r. We provide some more justification. Suppose the revoked users are uniformly distributed, i.e., r users are uniformly sampled one-by-one without replacement and revoked. Then the probability that the left subtree does not have any revoked user (and consequently the right subtree contains all of them) is

$$\begin{aligned}& \left(1-\frac{n/2}{n}\right)\left(1-\frac{n/2}{n-1}\right)\cdots\left(1-\frac{n/2}{n-r+1}\right) \\ = \quad & \left(1-\frac{1}{2}\right)\left(1-\frac{1}{2\left(1-\frac{1}{n}\right)}\right)\cdots\left(1-\frac{1}{2\left(1-\frac{r-1}{n}\right)}\right)\end{aligned}$$

The probability that the right subtree does not have any revoked user is also equal to this value. So, the total probability that the header generates a subset is twice this value. For practical applications of BE, the number of users n will be usually be much larger than the number of revoked users r and so the ratio r/n will be small. Then the above expression can be approximated by 2^{-r}. This is negligible even for values of r as small as 20 or so. Consequently, for practical situations, there will be almost no effect on the header length if the root level is not made special.

3.4 Storage Minimal Layering

For a given value of ℓ_0, let $\mathrm{SML}_0(\ell_0)$ denote a layering strategy $\boldsymbol{\ell}$ (or equivalently is given by the sequence of differences $\mathbf{d}$), such that $\mathsf{storage}_0(\boldsymbol{\ell})$ takes the minimum value among all possible layering strategies for a tree with ℓ_0 levels and having the root as a special level. Let $\#\mathrm{SML}_0(\ell_0)$ denote $\mathsf{storage}_0(\boldsymbol{\ell})$ where $\boldsymbol{\ell}$ is a storage minimal layering strategy. Similarly define $\mathrm{SML}_1(\ell_0)$ and $\#\mathrm{SML}_1(\ell_0)$ that exclude the root level from being special.

We describe a dynamic programming based algorithm to compute $\mathrm{SML}_0(\ell_0)$ (and subsequently $\mathrm{SML}_1(\ell_0)$). The idea of the algorithm is explained as follows. For a fixed value of ℓ_0, the number of layers e can vary from 1 to ℓ_0. The cases $e = 1$ and $e = \ell_0$ correspond to the SD scheme and in these two cases the user storage is known to be equal to $\ell_0(\ell_0 + 1)/2$. Let $\mathrm{SML}_0(e, \ell_0)$ denote a storage minimal layering *using exactly e layers.* Clearly, the following relation holds.

$$\#\mathrm{SML}_0(\ell_0) = \min_{1 \le e \le \ell_0} \#\mathrm{SML}_0(e, \ell_0). \tag{6}$$

Also,

$$\#\mathrm{SML}_0(e, \ell_0) = \min_{(\ell_0, \ldots, \ell_e)} \mathsf{storage}_0(\ell_0, \ell_1, \ldots, \ell_e) \tag{7}$$

where the minimum is over all possible layering strategies $(\ell_0, \ell_1, \ldots, \ell_e)$. Using (2)

$$\#\mathrm{SML}_0(e, \ell_0) = \min_{1 \le \ell_1 < \ell_0} \left(\ell_0 + \frac{(\ell_0 - \ell_1)(\ell_0 - \ell_1 - 1)}{2} + \#\mathrm{SML}_0(e - 1, \ell_1) \right). \tag{8}$$

This relation is the basis for the algorithm. Let Tab be an $\ell_0 \times \ell_0$ table such that $\mathsf{Tab}[e][\ell_0] = \#\mathrm{SML}_0(e, \ell_0)$. A simple $O(\ell_0^3)$ time dynamic programming algorithm can fill up this table as given in Algorithm 1.

Using (6) provides $\#\mathrm{SML}_0(\ell_0)$ as the minimum value in column number ℓ_0 of Tab. Note that the minimum may occur for more than one possible value of e. These values of ℓ_1 are reported during the computation. Let $\Lambda(e, \ell_0)$ be the list of all possible values of ℓ_1 for which (8) holds. The above method can be extended to generate all possible layering strategies for which user storage is minimized.

An SML_0 layering strategy $\boldsymbol{\ell}$ can be generated as follows. Start with $\boldsymbol{\ell}$ as the list containing only ℓ_0 and keep on appending in the following manner to obtain the complete sequence. Let e be one of the possibilities for which $\mathsf{Tab}[e][\ell_0]$ takes the minimum value; choose ℓ_1 as any one value from $\Lambda(e, \ell_0)$ and append to $\boldsymbol{\ell}$; choose ℓ_2 as any one value from $\Lambda(e - 1, \ell_1)$ and append to $\boldsymbol{\ell}$; continue until 0 is appended to the list. All SML_0 strategies can be generated by looping over all possible values of e, all possible values of ℓ_1, all possible values of ℓ_2 and so on.

ALGORITHM 1: Dynamic Programming Algorithm to find Tab

Input: ℓ_0.

Output: An $\ell_0 \times \ell_0$ table Tab where $\mathsf{Tab}[e][\ell]$ contains the value of $\#\mathrm{SML}_0(e,\ell)$.

for $\ell = 1$ *to* ℓ_0 **do**

  $\mathsf{Tab}[1][\ell] = \mathsf{Tab}[\ell][\ell] = \ell(\ell+1)/2$;

end

for $\ell = 2$ *to* ℓ_0 **do**

  for $e = 2$ *to* $\ell - 1$ **do**

    $\mathsf{Tab}[e][\ell] = \min\limits_{1 \le \ell_1 < \ell} \left(\ell + \frac{(\ell-\ell_1)(\ell-\ell_1-1)}{2} + \mathsf{Tab}[e-1][\ell_1] \right)$

  end

end

Once Tab is prepared, computing $\#\mathrm{SML}_1(\ell_0)$ using (5) is easy.

$$\begin{aligned}
&\#\mathrm{SML}_1(\ell_0) \\
&= \min_e \min_{\ell_1} \left(\#\mathrm{SML}_0(e-1,\ell_1) + \frac{(\ell_0-\ell_1)(\ell_0-\ell_1+1)}{2} \right) \\
&= \min_e \min_{\ell_1} \left(\mathsf{Tab}[e-1][\ell_1] + \frac{(\ell_0-\ell_1)(\ell_0-\ell_1+1)}{2} \right).
\end{aligned} \tag{9}$$

The first minimization is over the number of layers and the second minimization is over the value of the first special level. The possible corresponding layering strategies can also be easily recovered. It is to be noted that the $\mathrm{SML}_1(\ell_0)$ layerings are due to the minimization of the user storage by assuming the root to be at a non-special level. It can be seen from (8) and (9) that in an $\mathrm{SML}_0(\ell_0)$ layering, if the root is made non-special, it might not necessarily result in an $\mathrm{SML}_1(\ell_0)$ layering and vice versa.

Table 1 shows values of user storage for SML strategies for some ℓ_0. For comparison, we also show the storage requirements for the SD scheme and the e-HS layering strategy. Compared to the SD scheme, the e-HS layering strategy reduces the storage requirement very significantly (both asymptotically as well as in practical numbers). Compared to the e-HS scheme the value of $\#\mathrm{SML}_0(\ell_0)$ is slightly smaller and the value of $\#\mathrm{SML}_1(\ell_0)$ is about 18% to 24% lower for the newly suggested values of $\boldsymbol{\ell}$. So, given a value of ℓ_0, if the requirement is to minimize the user storage, then the SML strategies offer better alternatives. They also guarantee that using 2-way splitting of SD subsets with layering, further lowering of storage cannot be achieved.

The effect of $\mathrm{SML}_0(\ell_0)$ and $\mathrm{SML}_1(\ell_0)$ strategies on the average header length is also shown in Table 1. For computing the average header lengths, we have considered ten values of r equally spaced between $r_{\min}$ and $r_{\max}$. The reported values are the average header lengths of the different schemes normalized by the average header length of the SD scheme. As an example, the first value 1.69 corresponding to the row for e-HS and $\ell_0 = 28$ means that with $n = 2^{28}$ users out of which $r = 2^{10}$ are uniformly revoked, the average header length of the e-HS layering strategy is 1.69 times that of the SD scheme.

One may note the following points.

1. For a fixed ℓ_0, there may be more than one $\mathrm{SML}_0(\ell_0)$ (resp. $\mathrm{SML}_1(\ell_0)$) strategy which achieves storage of $\#\mathrm{SML}_0(\ell_0)$ (resp. $\#\mathrm{SML}_1(\ell_0)$). Table 2 gives the number of SML strategies for several values of ℓ_0. For $\ell_0 = 12$, Table 3 lists all possible $\mathrm{SML}_0(\ell_0)$ and $\mathrm{SML}_1(\ell_0)$ strategies for $\ell_0 = 12$. There, however, need not be a single layering strategy which minimizes expected header length for all possible values of r. Out of these, one would be interested in the layering that would give the minimum expected header length for most values of r under consideration. The SML strategies reported in Table 1 have this feature.

ℓ_0	$r_{\min}$	$r_{\max}$	scheme	special levels	storage	normalized header lengths for $(r_{\min}, \dots, r_{\max})$
12	2^2	2^6	SD	$12, 0$	78	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			e-HS	$12, 8, 4, 0$	**42**	(**1.69, 1.59, 1.56, 1.56, 1.57, 1.57, 1.57, 1.56, 1.55**, 1.53, 1.52)
			SML$_0$	$12, 8, 5, 3, 1, 0$	40	(**1.68, 1.57, 1.54, 1.54, 1.54, 1.55, 1.55, 1.54, 1.54**, 1.53, 1.52)
			SML$_1$	$8, 5, 3, 1, 0$	**32**	(**1.68, 1.57, 1.54, 1.54, 1.54, 1.55, 1.55, 1.54, 1.54**, 1.53, 1.52)
16	2^3	2^8	SD	$16, 0$	136	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			HS	$16, 12, 8, 4, 0$	**64**	(1.63, **1.65, 1.66, 1.64, 1.62, 1.60, 1.58, 1.57, 1.57, 1.56**)
			SML$_0$	$16, 11, 7, 4, 2, 1, 0$	61	(1.69, **1.60, 1.63**, 1.65, 1.65, 1.64, 1.63, 1.62, 1.60, 1.59)
			SML$_1$	$12, 8, 5, 3, 1, 0$	**50**	(1.63, **1.64, 1.65, 1.63, 1.60, 1.58, 1.57, 1.56, 1.55, 1.54**)
20	2^4	2^{10}	SD	$20, 0$	210	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			e-HS	$20, 15, 10, 5, 0$	**90**	(1.64, 1.72, 1.69, 1.66, **1.64, 1.62, 1.61, 1.61, 1.60, 1.60**)
			SML$_0$	$20, 15, 10, 6, 3, 1, 0$	85	(1.64, 1.72, 1.69, 1.66, **1.63, 1.62, 1.61, 1.60, 1.60, 1.60**)
			SML$_1$	$15, 10, 6, 3, 1, 0$	**70**	((1.64, 1.72, 1.69, 1.66, **1.63, 1.62, 1.61, 1.60, 1.60, 1.60**)
24	2^5	2^{12}	SD	$24, 0$	300	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			e-HS	$24, 19, 14, 9, 4, 0$	**116**	(1.62, 1.64, 1.62, 1.64, **1.67, 1.69, 1.71, 1.71, 1.72, 1.72**)
			SML$_0$	$24, 18, 12, 7, 3, 1, 0$	112	(1.65, 1.74, 1.70, 1.67, **1.65, 1.63, 1.63, 1.62, 1.62, 1.63**)
			SML$_1$	$18, 12, 8, 5, 3, 1, 0$	**94**	(1.65, 1.74, 1.69, 1.66, **1.63, 1.62, 1.61, 1.60, 1.60, 1.60**)
25	2^5	2^{12}	SD	$24, 0$	325	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			HS	$25, 20, 15, 10, 5, 0$	**125**	(1.62, 1.64, 1.62, 1.64, **1.67, 1.69, 1.71, 1.71, 1.72, 1.72**)
			SML$_0$	$25, 19, 13, 9, 6, 3, 1, 0$	119	(1.65, 1.74, 1.69, 1.66, **1.63, 1.62, 1.61, 1.60, 1.60, 1.60**)
			SML$_1$	$19, 13, 9, 6, 3, 1, 0$	**100**	(1.65, 1.74, 1.69, 1.66, **1.63, 1.62, 1.61, 1.60, 1.60, 1.60**)
28	2^6	2^{14}	SD	$28, 0$	406	(1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)
			e-HS	$28, 22, 16, 10, 5, 0$	**146**	(1.64, 1.65, **1.63, 1.66, 1.69, 1.71, 1.73, 1.74, 1.75, 1.75**)
			SML$_0$	$28, 21, 15, 10, 6, 3, 1, 0$	140	(1.65, 1.70, 1.65, **1.63, 1.62, 1.63, 1.64, 1.66, 1.67, 1.68**)
			SML$_1$	$22, 16, 11, 7, 4, 2, 0$	**119**	(1.64, 1.65, **1.62, 1.64, 1.67, 1.69, 1.70, 1.71, 1.72, 1.72**)

Table 1: Comparison of user storage and expected header lengths between e-HS LSD and SML. The tuples contain header lengths normalized with the SD header lengths corresponding to the values of r in $(r_{\min}, \dots, r_{\max})$ respectively.

ℓ_0	no. of SML$_0(\ell_0)$ layerings	no. of SML$_1(\ell_0)$ layerings
12	10	10
16	6	15
20	6	1
24	35	35
25	35	21
28	1	8

Table 2: The number of SML$_0(\ell_0)$ and SML$_1(\ell_0)$ layering strategies for various values of ℓ_0.

10 Special levels for SML$_0$(12)	10 Special levels for SML$_1$(12)
12,7,4,2,1,0	8,4,2,1,0
12,8,4,2,1,0	8,5,2,1,0
12,8,5,2,1,0	8,5,3,1,0
12,8,5,3,1,0	9,5,2,1,0
12,7,3,1,0	9,5,3,1,0
12,7,4,1,0	9,6,3,1,0
12,7,4,2,0	8,4,1,0
12,8,4,1,0	8,4,2,0
12,8,4,2,0	8,5,2,0
12,8,5,2,0	9,5,2,0

Table 3: List of SML$_0(\ell_0)$ and SML$_1(\ell_0)$ layering strategies denoted by the special levels for $\ell_0 = 12$.

2. For $\ell_0 = 32$, Tab has been computed and reported in Table I of the supplementary material. It gives the values of the minimum storage for every $1 \le \ell_0 \le 32$ and $1 \le e \le \ell_0$. For a particular ℓ_0 and e, it also gives the values of ℓ_1 for which (8) holds. As an example, we see that for $\ell_0 = 32$ and $e = 8$, $\#\mathrm{SML}_0(e, \ell_0) = 172$ and the values of ℓ_1 are 24 and 25. All possible $\mathrm{SML}_0(\ell_0)$ strategies for $1 \le \ell_0 \le 32$ can be obtained from this table and the $\mathrm{SML}_1(\ell_0)$ strategies can subsequently be found using (9).

3. As discussed earlier, if the root level is made non-special in an SML_0 strategy, it may not lead to an SML_1 strategy and vice versa. Table 3 shows that while the SML_0 strategy $\boldsymbol{\ell} = (12, 8, 4, 2, 1, 0)$ gives rise to an SML_1 strategy $\boldsymbol{\ell} = (8, 4, 2, 1, 0)$ by making the root level non-special, the SML_0 strategy $\boldsymbol{\ell} = (12, 7, 4, 2, 1, 0)$ does not. On the other hand, the SML_1 strategy $\boldsymbol{\ell} = (9, 5, 2, 1, 0)$ is not generated from an SML_0 strategy.

4. Extensive experimentation have shown that for practical values of r, there is no significant difference between the average header lengths of SML_0 and SML_1 strategies that differ at only the root being at a special level or not. For $\ell_0 = 12$ and 16, the reported SML_0 strategy with the root level made non-special turns out to be an SML_1 strategy (as reported in Table 1) with minimum expected header lengths. This supports the theoretical justification described before. However, for $\ell_0 = 20$, it turns out that making the root level of the SML_0 strategy non-special does not give rise to an SML_1 strategy. For $\ell_0 = 24$ and 28, it is again true that making the root level of the reported SML_0 strategy non-special gives rise to an SML_1 strategy. But there are other SML_1 strategies that further reduce the expected header lengths and hence we report those strategies in Table 1.

5. In general, the header length of the e-HS scheme is smaller than that of SML_0 and SML_1. This is somewhat expected, since user storage in SML is smaller. On the other hand, the user storage is not the only determining factor. The actual layering strategy also plays a role and in some cases it turns out that the average header length in SML turns out to be smaller than that in e-HS. We do not have an analytical justification for this. Intuitively, it appears that for the number of revoked users that have been considered, the SML assigns keys to SD subsets which are more probable to occur in the header. As a result, in such cases, we see that *both* user storage and average header length are reduced. These are marked in bold and are particularly noticeable for $\ell_0 = 24$ and $\ell_0 = 28$. In the context of AACS standard [4], SML_1 for $\ell_0 = 28$ is of particular significance.

3.5 Constrained Minimization of User Storage

From the viewpoint of minimizing communication bandwidth it is of interest to minimize the average header length. This is minimized when the number of keys is maximized which happens for the SD scheme, i.e., when all the levels are considered to be special levels or there is only a single layer. Taking the average header length for the SD scheme as a benchmark, one may ask the question as to how much the user storage can be reduced from that required by the SD scheme without significantly increasing the corresponding values for the average header length? The expression for the average header length (as can be derived from (11), (13) and Proposition 2 given later) is rather complicated and it appears quite impossible to have an analytical solution to this question. Instead, we use our average header length computation program (developed in Section 4.3) to study this behaviour for concrete practical values of n, r and layering strategies $\boldsymbol{\ell}$. It turns out that it is indeed possible to significantly reduce the user storage values with minimal increase in the average header length values.

Our approach is the following. The increase in header length due to layering occurs because of the fact that certain SD subsets are split into two. If we can avoid making too many splits, then we can ensure that the header length does not increase by too much in comparison to the SD scheme. Consider an SD subset of the form $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ where node i is at level ℓ. We say that this subset is generated from the node i. Now, consider the expected number of SD subsets that will be generated from all the nodes at level ℓ. If this number is 'large', then we make the level ℓ special. This ensures that SD subsets originating level ℓ will not be split. Overall, the strategy is to ensure that SD subsets originating from levels which contribute most to the header are not split. This mitigates the effect of splits.

Suppose there are n users and r of them are revoked. In [5] it has been shown that the probability that a particular node at level ℓ generates a subset in the header is $2(\eta_r(n, 2^{\ell-1}) - \eta_r(n, 2 \times 2^{\ell-1}) - \eta_r(n, 3 \times 2^{\ell-1}) + \eta_r(n, 4 \times 2^{\ell-1}))$ where $\eta_r(n, x) = (1 - x/n)(1 - x/(n-1)) \cdots (1 - x/(n-r+1))$ if $n > r-1$ else 0. Since there are $2^{\ell_0-\ell}$ nodes at level ℓ, the expected number of subsets arising from all nodes at level ℓ is

$$\begin{aligned} 2^{\ell_0-\ell+1}&(\eta_r(n, 2^{\ell-1}) - \eta_r(n, 2 \times 2^{\ell-1}) \\ &-\eta_r(n, 3 \times 2^{\ell-1}) + \eta_r(n, 4 \times 2^{\ell-1})). \end{aligned} \tag{10}$$

This expression gives the expected contribution of a level to the header size for a given r.

For a fixed n and r, one can consider the problem of finding ℓ for which (10) is maximized. Analytically, this seems to be very difficult to do. Instead we have done extensive experimentation. Empirical values suggest that the maximum occurs for some level $\ell \leq \ell_0 - \lfloor \log_2 r \rfloor$. Also, for $\ell > \ell_0 - \lfloor \log_2 r \rfloor$, the value of (10) is quite small.

Based on this empirical evidence we suggest the following layering strategy.

- Make level $\ell_0 - \lfloor \log_2 r \rfloor$ special. Level 0 is also special.
- No level $0 < \ell < \ell_0 - \lfloor \log_2 r \rfloor$ is made special. In terms of user storage and expected header length this is equivalent to making all levels $\ell < \ell_0 - \lfloor \log_2 r \rfloor$ to be special.
- The root level is not made special.
- At most one level that is midway between ℓ_0 and $\ell_0 - \lfloor \log_2 r \rfloor$ is made special. While this does not significantly affect header size, it can reduce the storage requirement.

We call this the *constrained minimization layering (CML)* strategy. This strategy will ensure that if $\ell \leq \ell_0 - \lfloor \log_2 r \rfloor$, then no SD subset generated from level ℓ or below will be split. Splits will occur only for SD subsets originating from levels above ℓ. But, the expected number of such subsets is small and so, splits will occur only for a small number of SD subsets.

One issue with this strategy is that the value of r will not be known a priori while the layering scheme will have to be decided upon during the design phase itself. A way out is to make an assumption about the minimum number of revoked users that will occur in the steady state operation of the BE scheme. For example, in AACS with 2^{28} users one may assume that in the steady state at least 2^{10} users will be revoked due to equipment piracy problems.

Suppose that $r_{\min}$ is the minimum number of users that will be revoked during each broadcast. The above layering strategy is used with $r_{\min}$. Suppose now that during a broadcast, the number of users r that is actually revoked is greater than $r_{\min}$. Then from our empirical evidence the level for which the average header length is maximized will be $\ell_0 - \lfloor \log_2 r \rfloor$. Since this value is less than $\ell_0 - \lfloor \log_2 r_{\min} \rfloor$, none of the subsets generated from this level will be split. So, the feature of not splitting a large number of SD subsets is still retained.

Table 4 shows a comparison between the SD scheme, the e-HS layering scheme and a constrained minimization layering scheme as described above, in terms of both their user storage requirement and the expected header length normalized with respect to the SD scheme. The average header length depends on the number r of revoked users. So, for a given $n = 2^{\ell_0}$, we computed the expected header lengths for 10 equispaced values of r between and including $r_{\min}$ and $r_{\max}$. The values in the table illustrate the point that compared to the SD scheme, the constrained minimization layering scheme substantially reduces the user storage with a small increase in the average header length.

The layering scheme is designed assuming that the number of revoked users is at least $r_{\min}$. What happens if the number of revoked users in an actual broadcast is smaller than $r_{\min}$? Clearly, we cannot expect the average header length to still be almost equal to that of the SD scheme. This effect is shown for some values of r in

ℓ_0	$r_{\min}$	$r_{\max}$	scheme	special levels	storage	normalized header lengths for $(r_{\min}, \ldots, r_{\max})$
12	2^2	2^6	SD	$12, 0$	78	$(1, \ldots, 1)$
			e-HS	$12, 8, 4, 0$	42	$(1.69, 1.59, 1.56, 1.56, 1.57, 1.57, 1.57, 1.56, 1.55, 1.53, 1.52)$
			CML	$10, 0$	58	$(1.15, 1.01, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
16	2^6	2^8	SD	$16, 0$	136	$(1, \ldots, 1)$
			HS	$16, 12, 8, 4, 0$	64	$(1.66, 1.64, 1.62, 1.61, 1.59, 1.58, 1.58, 1.57, 1.57, 1.56)$
			CML	$10, 0$	76	$(1.14, 1.08, 1.05, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$
20	2^8	2^{10}	SD	$20, 0$	210	$(1, \ldots, 1)$
			e-HS	$20, 15, 10, 5, 0$	90	$(1.68, 1.66, 1.64, 1.63, 1.62, 1.61, 1.61, 1.60, 1.60, 1.60)$
			CML	$16, 12, 0$	110	$(1.14, 1.08, 1.04, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$
24	2^{10}	2^{12}	SD	$24, 0$	300	$(1, \ldots, 1)$
			e-HS	$24, 19, 14, 9, 4, 0$	116	$(1.63, 1.64, 1.66, 1.68, 1.69, 1.71, 1.71, 1.72, 1.72, 1.72)$
			CML	$19, 14, 0$	149	$(1.14, 1.08, 1.04, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$
25	2^{10}	2^{12}	SD	$25, 0$	325	$(1, \ldots, 1)$
			e-HS	$25, 20, 15, 10, 5, 0$	125	$(1.63, 1.64, 1.66, 1.68, 1.69, 1.71, 1.71, 1.72, 1.72, 1.72)$
			CML	$20, 15, 0$	165	$(1.14, 1.08, 1.04, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$
28	2^{10}	2^{14}	SD	$28, 0$	406	$(1, \ldots, 1)$
			e-HS	$28, 22, 16, 10, 5, 0$	146	$(1.69, 1.63, 1.64, 1.67, 1.69, 1.72, 1.73, 1.74, 1.75, 1.75)$
			CML	$23, 18, 0$	219	$(1.14, 1.08, 1.04, 1.03, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$

Table 4: Comparison of user storage and average header length for SD, e-HS LSD and the constrained minimization layering. The tuples contain header lengths normalized with the SD header lengths corresponding to the values of r in $(r_{\min}, \ldots, r_{\max})$ respectively.

Table 5. Again the values of the average header length are normalized by that of the corresponding SD scheme. For comparison, we have also provided the average header lengths of the e-HS layering strategy. It is to be noted that the expected header lengths of the CML scheme are mostly better than the e-HS scheme. As an example, for $n = 2^{24}$, for $r > 6$, the CML strategy gives smaller expected header lengths than the e-HS layering strategy. Table 5 shows that for any value of n, the CML strategy leads to smaller expected header lengths for all $r > 15$.

To summarize, the constrained minimization layering strategy requires significantly lesser user storage than the SD scheme. In terms of the expected header length, it is as good as the SD scheme for $r \geq r_{\min}$. If $r < r_{\min}$, then it is better than e-HS layering but inferior to the SD scheme. It is to be noted that if r is small, then the absolute size of the header itself is not too large. As a result, the effective transmission overhead of the scheme will never be too high compared to the actual body of the message.

4 Header Length

The main point of the discussion in this section is to obtain an efficient algorithm for computing the expected header length for the layered SD schemes including the LSD scheme. The algorithm we obtain works for all possible values of the number of users. To ensure this, we first need to extend the scheme to handle arbitrary number of users. For the SD scheme, this was done in [5] by using the notion of complete binary trees. Here, we extend the scheme of [5] to handle layering as well.

4.1 Tackling Arbitrary Number of Users

In [3] and [6], the number of users has been taken to be a power of two, i.e., $n = 2^{\ell_0}$. One has to consider dummy users in the system to make the number of users a power of two. The inclusion of dummy users (considered revoked or privileged) increase the expected header length in the system. Hence, this is not always convenient as has been argued in details in [5].

By modifying the structure of the tree, it is possible to handle arbitrary number of users. This modification is based on the notion of complete binary trees. These are trees where the leaf nodes are at the last and maybe

ℓ_0	$r_{\min}$	scheme	special levels	storage	header lengths normalized with the SD scheme
					$r = (1, \mathbf{2}, 3, 4)$
12	2^2	e-HS	$12, 8, 4, 0$	42	$(1.00, \mathbf{1.74}, 1.72, 1.69)$
		CML	$10, 0$	58	$(2.00, \mathbf{1.50}, 1.26, 1.15)$
					$r = (2, 4, 6, 8, 10, \mathbf{12}, 14, 16, 18, 20)$
16	2^6	HS	$12, 8, 4, 0$	64	$(1.75, 1.70, 1.66, 1.63, 1.61, \mathbf{1.60}, 1.60, 1.60, 1.60, 1.61)$
		CML	$10, 0$	76	$(1.78, 1.74, 1.70, 1.66, 1.63, \mathbf{1.59}, 1.56, 1.53, 1.50, 1.47)$
					$r = (2, \mathbf{4}, 6, 8, 10, 12, 14, 16, 18, 20)$
20	2^8	e-HS	$20, 15, 10, 5, 0$	90	$(1.77, \mathbf{1.75}, 1.72, 1.70, 1.68, 1.66, 1.65, 1.64, 1.63, 1.63)$
		CML	$16, 12, 0$	110	$(1.77, \mathbf{1.69}, 1.64, 1.61, 1.59, 1.57, 1.56, 1.56, 1.56, 1.56)$
					$r = (2, 4, 6, \mathbf{8}, 10, 12, 14, 16, 18, 20)$
24	2^{10}	e-HS	$24, 19, 14, 9, 4, 0$	116	$(1.77, 1.75, 1.72, \mathbf{1.70}, 1.68, 1.66, 1.65, 1.64, 1.63, 1.63)$
		CML	$19, 14, 0$	149	$(1.79, 1.75, 1.72, \mathbf{1.69}, 1.67, 1.65, 1.64, 1.63, 1.62, 1.61)$
					$r = (2, 4, 6, \mathbf{8}, 10, 12, 14, 16, 18, 20)$
25	2^{10}	e-HS	$25, 20, 15, 10, 5, 0$	125	$(1.77, 1.75, 1.72, \mathbf{1.70}, 1.68, 1.66, 1.65, 1.64, 1.63, 1.63)$
		CML	$20, 15, 0$	165	$(1.79, 1.75, 1.72, \mathbf{1.69}, 1.67, 1.65, 1.64, 1.63, 1.62, 1.61)$
					$r = (2, \mathbf{4}, 6, 8, 10, 12, 14, 16, 18, 20)$
28	2^{10}	e-HS	$28, 22, 16, 10, 5, 0$	146	$(1.79, \mathbf{1.78}, 1.76, 1.74, 1.73, 1.72, 1.71, 1.70, 1.69, 1.68)$
		CML	$23, 18, 0$	219	$(1.79, \mathbf{1.75}, 1.72, 1.69, 1.67, 1.65, 1.64, 1.63, 1.62, 1.61)$

Table 5: Comparison of average header length for $r < r_{\min}$ between e-HS layering strategy and the constrained minimization layering strategy.

the second last levels. The last level has all its nodes to the left side. An example of a complete subtree accommodating 13 users is shown in Figure 5. In this case $\ell_0 = 4$ and choosing $d = 2$ gives two layers and three special levels as shown in the figure. When the number of users is a power of two, the corresponding tree is called a full binary tree. This difference in terminology between full and complete has been taken from the literature on data structures. We explain some terminology with respect to Figure 5. The left and the right subtrees of node 3 are the subtrees rooted at nodes 7 and 8 respectively. The sibling subtree of node 3 is the subtree rooted at node 4. The only non-full subtrees are those rooted at nodes 0, 2 and 5. We call the path labelled by the nodes 0, 2 and 5 to be the *dividing path*.

In general given n with $2^{\ell_0 - 1} < n \leq 2^{\ell_0}$, it is possible to accommodate n users as the leaves of a complete binary tree with n leaves. The root node is at level ℓ_0. The leaves and hence the users are either at level 0 or at level 1. Suppose the sequence of special levels is $\boldsymbol{\ell} = (\ell_0, \ldots, \ell_e)$. For users at level 0, the storage requirement is $\mathsf{storage}_0(\boldsymbol{\ell})$ while for users at level 1, the storage requirement is $\mathsf{storage}_0(\boldsymbol{\ell}) - (e + p - 2)$ where p is the number of levels in the bottom-most layer. This reduction is due to the fact that these users need to store one less label for each special level above it and for each level in its last layer. The distribution of labels using the PRG is done as usual.

During a broadcast, the actual header generation is done in much the same way. First, as in the SD scheme, the set of non-revoked users is covered exactly by subsets of the form $\mathcal{S}^i \setminus \mathcal{S}^j$ where i is a node in the tree and j is a node in the subtree rooted at i. If i is at a non-special level and j is not in the same layer as i, then this set is further split into $\left(\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}\right) \cup \left(\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}\right)$ where k is the first node appearing at a special level on the path from i to j.

Complications for complete but non-full trees arise due to the following reason. For the internal nodes lying on the dividing path, the subtree rooted at it may not be full. A node not on the dividing path and at level ℓ is the root of a subtree having either 2^{ℓ} leaves or $2^{\ell-1}$ leaves accordingly as whether the node is to the left or to the right of the dividing path. As an example, in Figure 5, nodes 3, 4, 5 and 6 are at level 2. Node 5 is on the dividing path and the subtree rooted at node 5 is non-full; nodes 3 and 4 are to the left of 5 and are the roots of subtrees having $2^2 = 4$ leaves; node 6 is to the right of node 5 and the subtree rooted at 6 has 2 leaves.

The LSD scheme is based on full binary trees and this extension to complete binary trees gives rise to the *complete tree layered subset difference (CTLSD) scheme*. The LSD scheme had improved upon the SD scheme by

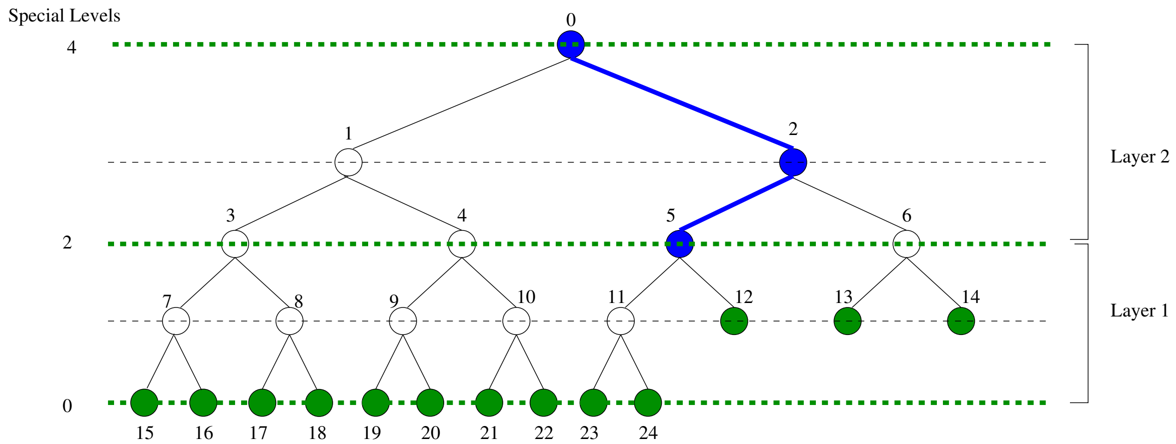

Figure 5: A complete tree with 13 leaf nodes. The levels 0, 2 and 4 are special levels and hence there are two layers. The nodes 0, 2 and 5 are roots of non-full complete subtrees and hence they lie on the dividing path.

reducing the user storage at the cost of almost double the transmission overhead. The CTLSD scheme subsumes all these schemes by accommodating arbitrary number of users and allowing appropriate choices of the layering strategy $\boldsymbol{\ell}$ for specific applications.

4.2 Maximum Header Length

Before considering the expected header length, we state the following bound on the worst case header length. The proof is given in the supplementary material.

Proposition 1. *The maximum header length in the CTLSD scheme for n users out of which r are revoked is* $\min\left(4r-2, \left\lceil \frac{n}{2} \right\rceil, n-r\right)$. *If the root is a special level, then the bound is* $\min\left(4r-3, \left\lceil \frac{n}{2} \right\rceil, n-r\right)$.

4.3 Expected Header Length

Assume that the layering strategy is given by $\boldsymbol{\ell} = (\ell_0, \ell_1, \ldots, \ell_e)$. Additionally, the information as to whether the root level is or is not special is also provided as a bit β. If $\beta = 0$, then the root node is special and if $\beta = 1$, the root node is not special. So, $(\boldsymbol{\ell}, \beta)$ provides complete information about the layering strategy. For compactness, we denote this as $\boldsymbol{\ell}_\beta$.

The expected header length is computed under the following random experiment. Out of n users, a set of r users are chosen uniformly at random and these users are revoked. The corresponding header length is then a random variable and let $Y_{n,r}$ denote this header length. We are interested in $E[Y_{n,r}]$. Due to the random revocation of the users, for each internal node i, there arise three possibilities: $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is added to the header; $\left(\mathcal{S}^{(i)} \setminus \mathcal{S}^{(k)}\right) \cup \left(\mathcal{S}^{(k)} \setminus \mathcal{S}^{(j)}\right)$ is added to the header; or nothing is added to the header. So, corresponding to node i, either 0 or 1 or 2 subsets are added to the header. Denote this number by $Y^i_{n,r}$. Then $Y_{n,r} = \sum Y^i_{n,r}$ where the sum is taken over all internal nodes i.

Computing this directly is not convenient. So, we simplify it further. Let $X^i_{n,r}$ be a binary valued random variable which takes the value 1 if and only if there is at least one subset generated from i and let $Z^i_{n,r}$ be another binary valued random variable which takes the value 1 if and only if there are exactly two subsets generated from i. (Note that if i is at a special level, then the probability $Z^i_{n,r} = 1$ is 0.) Then it follows that $Y^i_{n,r} = X^i_{n,r} + Z^i_{n,r}$. The reasoning is as follows. If i generates no subset, then both sides are zero; if exactly one subset is generated, then $Y^i_{n,r}$ and $X^i_{n,r}$ are both 1 but, $Z^i_{n,r}$ is 0; if exactly two subsets are generated then $Y^i_{n,r}$ is 2 and both $X^i_{n,r}$

and $Z^i_{n,r}$ are 1. By linearity of expectation, we have

$$\begin{aligned} E[Y_{n,r}] &= E\left[\sum Y^i_{n,r}\right] = \sum E\left[X^i_{n,r} + Z^i_{n,r}\right] \\ &= \sum E\left[X^i_{n,r}\right] + \sum E\left[Z^i_{n,r}\right]. \end{aligned} \tag{11}$$

The sum is over all internal nodes i of the tree. The quantity $\sum X^i_{n,r}$ is exactly the expected header length obtained using the SD algorithm. This is because i generates at least one subset if and only if the SD algorithm results in i generating a subset. Let $X_{n,r} = \sum X^i_{n,r}$ and $Z_{n,r} = \sum Z^i_{n,r}$. So,

$$E\left[Y_{n,r}\right] = E\left[X_{n,r}\right] + E\left[Z_{n,r}\right]. \tag{12}$$

An algorithm for computing $E[X_{n,r}]$ has been already developed in [5]. So, it only remains to determine $E[Z_{n,r}]$.

Given n and a layering sequence $\boldsymbol{\ell}_\beta$ we define the set $\mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$ to consist of pairs of nodes (i, j) such that i is not at a special level and j is in the subtree rooted at i but not in the same layer as i. So, whenever an SD subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is such that $(i, j) \in \mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$, it is split into two subsets. If i is at level ℓ, then there are at most $\ell - 1$ values of level for j such that (i, j) is in $\mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$.

Let i be at a non-special level and let j be not in the same layer as i. Define the binary valued random variable $W^{i,j}_{n,r}$ to take the value 1 if and only if the SD algorithm returns the subset $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ to the header, in which case the LSD algorithm will split this subset into two sets. So, we have $Z^i_{n,r} = \sum_{(i,j) \in \mathsf{SubsetsForSplit}(n,\boldsymbol{\ell}_\beta)} W^{i,j}_{n,r}$. Again by linearity of expectation, the task reduces to computing $E[W^{i,j}_{n,r}]$. Since this is a binary valued random variable, $E[W^{i,j}_{n,r}] = \Pr[W^{i,j}_{n,r} = 1]$. So,

$$\begin{aligned} E[Z_{n,r}] &= \sum_i E[Z^i_{n,r}] \\ &= \sum_i \sum_{(i,j) \in \mathsf{SubsetsForSplit}(n,\boldsymbol{\ell}_\beta)} \Pr[W^{i,j}_{n,r} = 1]. \end{aligned} \tag{13}$$

Here the first sum is over all nodes i at non-special levels. For a fixed i and j, we show how to compute $\Pr[W^{i,j}_{n,r} = 1]$. To do this, we need to characterize the event $W^{i,j}_{n,r} = 1$ for a pair $(i, j) \in \mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$. This event occurs if and only if the following conditions hold.

- Node i is either the root (in which case it does not have any sibling tree) or the sibling tree of i has at least one revoked user among its leaves.
- Either j is a leaf and is revoked or both subtrees of j have at least one revoked user among its leaves.
- There are no revoked users in the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$.

Define the following events:

1. R^j_{lt}: there is at least one revoked user in the left subtree of j;
2. R^j_{rt}: there is at least one revoked user in the right subtree of j;
3. R^i_{sb}: there is at least one revoked user in the sibling subtree of i;
4. $R^{i,j}_{rm}$: there is at least one revoked user in the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$.

Let $(i,j) \in \mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$. Suppose i is not the root. If j is not a leaf node, the event $W_{n,r}^{i,j} = 1$ is equivalent to the event $R_{sb}^{i} \wedge \overline{R_{rm}^{i,j}} \wedge R_{lt}^{j} \wedge R_{rt}^{j}$. If j is a leaf node, the event $W_{n,r}^{i,j} = 1$ is equivalent to the event $R_{sb}^{i} \wedge \overline{R_{rm}^{i,j}}$. Now suppose i is the root and is not special (i.e., $\beta = 1$). If j is not a leaf, then the event $W_{n,r}^{i,j} = 1$ is equivalent to $\overline{R_{rm}^{i,j}} \wedge R_{lt}^{j} \wedge R_{rt}^{j}$. If j is a leaf, then this can happen only if there is a single revoked user. So, for $r = 1$, the probability of $W_{n,r}^{i,j} = 1$ is 1 and for $r \geq 2$, the probability of $W_{n,r}^{i,j} = 1$ is 0.

Let λ_i (resp. λ_j; λ_s) be the number of leaves in the subtree rooted at i (resp. j; the sibling subtree of i). Similarly, let λ_{2j+1} and λ_{2j+2} respectively be the number of leaves in the left and right subtrees of j. So, $\lambda_j = \lambda_{2j+1} + \lambda_{2j+2}$. The number of leaves in the set $\mathcal{S}^{(i)} \setminus \mathcal{S}^{(j)}$ is $\lambda_i - \lambda_j$. Note that since we are dealing with arbitrary number of users, the subtrees that are being considered are not necessarily full. So, the values of the λ's are not necessarily powers of two.

Fix t users and consider the probability $\eta_r(n,t)$ that in the random experiment none of these t users have been chosen. Recall that the random experiment is to choose r users uniformly and without replacement from the set of n users. As discussed earlier

$$\eta_r(n,t) = \left(1 - \frac{t}{n}\right)\left(1 - \frac{t}{n-1}\right)\cdots\left(1 - \frac{t}{n-r+1}\right).$$

This makes it convenient to express the probability that none among a set of users of certain size is revoked. For example, the probability of $\overline{R_{lt}^{j}}$ is $\eta_r(n, \lambda_{2j+1})$. Similarly, the probability of the event $\overline{R_{lt}^{j}} \wedge \overline{R_{rm}^{i,j}}$ is $\eta_r(n, \lambda_{2j+1} + \lambda_i - \lambda_j) = \eta_r(n, \lambda_i - \lambda_{2j+2})$. Such calculations will be used in what follows.

Proposition 2. *Let i and j be nodes such that $(i,j) \in \mathsf{SubsetsForSplit}(n, \boldsymbol{\ell}_\beta)$.*

- *If i is the root and j is a leaf, then $\Pr[W_{n,r}^{i,j} = 1] = 1$ if $r = 1$ and $\Pr[W_{n,r}^{i,j} = 1] = 0$ if $r \geq 2$.*
- *If i is the root and j is not a leaf, then*

$$\begin{aligned}\Pr[W_{n,r}^{i,j} = 1] = {} & \eta_r(n, \lambda_i - \lambda_j) - \eta_r(n, \lambda_{2j+1} + \lambda_i - \lambda_j) \\ & - \eta_r(n, \lambda_{2j+2} + \lambda_i - \lambda_j) \\ & + \eta_r(n, \lambda_{2j+1} + \lambda_{2j+2} + \lambda_i - \lambda_j).\end{aligned} \tag{14}$$

- *If i is not the root and j is a leaf, then*

$$\Pr[W_{n,r}^{i,j} = 1] = \eta_r(n, \lambda_i - \lambda_j) - \eta_r(n, \lambda_s + \lambda_i - \lambda_j). \tag{15}$$

- *If i is not the root and j is not a leaf, then*

$$\begin{aligned}\Pr[W_{n,r}^{i,j} = 1] = {} & \eta_r(n, \lambda_i - \lambda_j) - \eta_r(n, \lambda_s + \lambda_i - \lambda_j) \\ & - \eta_r(n, \lambda_{2j+1} + \lambda_i - \lambda_j) \\ & - \eta_r(n, \lambda_{2j+2} + \lambda_i - \lambda_j) \\ & + \eta_r(n, \lambda_s + \lambda_{2j+1} + \lambda_i - \lambda_j) \\ & + \eta_r(n, \lambda_s + \lambda_{2j+2} + \lambda_i - \lambda_j) \\ & + \eta_r(n, \lambda_{2j+1} + \lambda_{2j+2} + \lambda_i - \lambda_j) \\ & - \eta_r(n, \lambda_s + \lambda_{2j+1} + \lambda_{2j+2} + \lambda_i - \lambda_j).\end{aligned} \tag{16}$$

The proof of this proposition is given in the supplementary material.

Algorithm to compute $Z_{n,r}$: For any fixed $(i,j) \in \mathsf{SubsetsForSplits}(n, \boldsymbol{\ell}_\beta)$, Theorem 2 provides a method for computing $\Pr[W_{n,r}^{i,j} = 1]$. Each of the η expressions can be computed using r multiplications and since there are a constant number of η's, the value of $\Pr[W_{n,r}^{i,j} = 1]$ can be computed using $O(r)$ multiplications. Using (13) this immediately gives a method for computing $Z_{n,r}$. Doing this directly, however, is not very efficient. The first sum in (13) is over all possible nodes i and the second sum is over the relevant j which are paired with i. Since the number of nodes is $O(n)$, a direct computation will lead to an algorithm whose running time is $O(rn^2)$.

This can be significantly improved. To explain the idea, first consider n to be a power of two so that the tree is a full binary tree. Fix a non-special node i and consider all possible j for which the second sum in (13) has to be evaluated. From the expression for $\Pr[W_{n,r}^{i,j} = 1]$ it is easy to note that for a fixed (n and r and) i, the value of $\Pr[W_{n,r}^{i,j} = 1]$ is determined only by the number of leaves in the subtree rooted at j and consequently the number of leaves in the left and the right subtrees of j. Since the tree is full, these values depend only on the value of the level of node j. So, for each appropriate level below i, one can compute the value of $\Pr[W_{n,r}^{i,j} = 1]$ for one particular j at that level and then multiply by the number of nodes in the subtree rooted at i at the level of j. As a result, the second sum in (13) can be computed in $O(r \log \lambda_i)$ time where λ_i is the number of leaves in the subtree rooted at i so that $\log \lambda_i$ is the level number of i. Since $\lambda_i \leq n$, the second sum in (13) can be computed using $O(r \log n)$ time.

Consider now the first sum in (13) (and still assume that n is a power of two). Again, it is easy to note that the value of $E[Z_{n,r}^i]$ is determined by the value of the level number of i. So, for each appropriate level, one can compute $E[Z_{n,r}^i]$ for one i and then multiply by the number of nodes at that level. As a result, computing $E[Z_{n,r}]$ requires a total of $O(r \log^2 n)$ multiplications.

If n is not a power of two, then the tree is a complete but, non-full tree and we need to revise the above description. The idea that all nodes at the same level contribute the same value does not hold any more. This is because the number of leaves in the subtrees rooted at nodes at the same level can be different. There is however, a way out which is based on the idea of the dividing path. One may recollect that the dividing path joins all nodes that are roots of non-full subtrees. All nodes at the same level and on the same side of the dividing path have the same number of leaf nodes. So, for each level, we compute separately for three cases: for nodes to the left of the dividing path; for the node on the dividing path; and for nodes to the right of the dividing path. For nodes at the same level and on the same side of the dividing path, we compute $\Pr[W_{n,r}^{i,j} = 1]$ once and multiply by the number of nodes satisfying this condition. Similarly the computation of $E[Z_{n,r}^i]$ is carried out. Overall, the complexity of the algorithm is still $O(r \log^2 n)$.

There is one complication that we have not explained. This is the problem of characterizing the dividing path and counting the number of nodes at the same level and on the same side of the dividing path. It turns out that given the value of n, this can always be done. The details are provided in [5] and so are omitted here. We have incorporated these in our implementation of the algorithm to compute expected header length given any value of n and r.

The expected header length of the CTLSD method is $E[Y_{n,r}]$. As given in (12), this quantity is equal to the sum of $E[X_{n,r}]$ and $E[Z_{n,r}]$. We have shown that $E[Z_{n,r}]$ can be computed in $O(r \log^2 n)$ time. The quantity $E[X_{n,r}]$ is the expected header length of the CTSD scheme and can be computed in $O(r \log n)$ time [5]. So, the overall complexity of the algorithm is $O(r \log^2 n)$.

Table 6 provides some examples of running the algorithm for computing expected header length for non-full trees using the CTSD and the CTLSD schemes. The chosen values of r are 10 equispaced values between $r_{\min}$ and $r_{\max}$ for the respective n. The CTLSD method is run by adopting the constrained minimization layering strategy where all levels including and below $\ell_0 - \lfloor \log_2 r_{\min} \rfloor$ are considered to be in one layer. The expected header length of the CTLSD method is almost similar to the CTSD scheme while the user storage requirement is a little more than half of the CTSD scheme. Hence, with an assumption on the minimum number of revoked users, the CTLSD scheme with the constrained minimization layering strategy would be the more practical choice.

n	scheme	special layers	storage	$r_{\min}$	$r_{\max}$	header length normalized by CTSD
10^3	CTSD	10,0	55	2^2	2^5	$(1,\ldots,1)$
	CTLSD	8,0	**39**	2^2	2^5	$(1.09, 1.02, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
10^4	CTSD	14,0	105	2^4	2^7	$(1,\ldots,1)$
	CTLSD	10,0	**65**	2^4	2^7	$(1.04, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
10^5	CTSD	17,0	153	2^6	2^8	$(1,\ldots,1)$
	CTLSD	11,0	**87**	2^6	2^8	$(1.08, 1.04, 1.02, 1.01, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
10^6	CTSD	20,0	210	2^8	2^{10}	$(1,\ldots,1)$
	CTLSD	16,12,0	**110**	2^8	2^{10}	$(1.13, 1.07, 1.04, 1.02, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$
10^7	CTSD	24,0	300	2^{10}	2^{12}	$(1,\ldots,1)$
	CTLSD	19,14,0	**149**	2^{10}	2^{12}	$(1.04, 1.02, 1.01, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
10^8	CTSD	27,0	378	2^{10}	2^{13}	$(1,\ldots,1)$
	CTLSD	22,17,0	**200**	2^{10}	2^{13}	$(1.08, 1.04, 1.02, 1.01, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00)$
10^9	CTSD	30,0	465	2^{10}	2^{15}	$(1,\ldots,1)$
	CTLSD	25,20,0	**260**	2^{10}	2^{15}	$(1.12, 1.07, 1.04, 1.02, 1.01, 1.01, 1.00, 1.00, 1.00, 1.00)$

Table 6: Comparison of the storage and the expected header lengths for the CTSD and the CTLSD (with constrained minimization layering) schemes.

Since the CTLSD scheme subsumes the HS LSD and the e-HS LSD schemes, this algorithm computes the expected header length for these schemes too. In [6], it was mentioned that the expected header length for their layering scheme, i.e; HS layering is around $2r$. As we have seen earlier, by suitably placing the special levels, this can be brought down significantly to about the expected header length of the SD scheme. On the other hand, for the (e-)HS scheme, the expected header length can also be somewhat larger than $2r$. For example, for $l_0 = 28$ and $r = 2$, the expected header length is $2.23r$.

5 Conclusion

In this work, we have suggested new layering strategies for the SD scheme. At one end we have shown that it is possible to decrease the user storage below that obtained by Halevy and Shamir [6]. At the other end, we have shown that it is possible to attain header length very close to that of the SD scheme while still requiring a significantly smaller number of keys. The LSD scheme is extended to handle arbitrary number of users leading to the CTLSD scheme. We have obtained an efficient algorithm to compute the expected header length in the CTLSD scheme. Our analysis of different scenarios is made possible by using this algorithm.

Acknowledgement

We would like to thank the anonymous reviewers for their comments and suggestions which has helped to improve the paper.

References

[1] Shimshon Berkovits, "How to broadcast a secret," in *EUROCRYPT*, Donald W. Davies, Ed. 1991, vol. 547 of *Lecture Notes in Computer Science*, pp. 535–541, Springer.

[2] Amos Fiat and Moni Naor, "Broadcast encryption," in *CRYPTO*, Douglas R. Stinson, Ed. 1993, vol. 773 of *Lecture Notes in Computer Science*, pp. 480–491, Springer.

[3] Dalit Naor, Moni Naor, and Jeffery Lotspiech, "Revocation and tracing schemes for stateless receivers," in *CRYPTO*, Joe Kilian, Ed. 2001, vol. 2139 of *Lecture Notes in Computer Science*, pp. 41–62, Springer.

[4] AACS, "Advanced Access Content System, `http://www.aacsla.com`," .

[5] Sanjay Bhattacherjee and Palash Sarkar, "Complete tree subset difference broadcast encryption scheme and its analysis," *Designs, Codes and Cryptography*, to appear.

[6] Dani Halevy and Adi Shamir, "The LSD broadcast encryption scheme," in *CRYPTO*, Moti Yung, Ed. 2002, vol. 2442 of *Lecture Notes in Computer Science*, pp. 47–60, Springer.

[7] Douglas R. Stinson, "On some methods for unconditionally secure key distribution and broadcast encryption," *Des. Codes Cryptography*, vol. 12, no. 3, pp. 215–243, 1997.

[8] Douglas R. Stinson and Ruizhong Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes," *SIAM J. Discrete Math.*, vol. 11, no. 1, pp. 41–53, 1998.

[9] E. C. Park and Ian F. Blake, "On the mean number of encryptions for tree-based broadcast encryption schemes," *J. Discrete Algorithms*, vol. 4, no. 2, pp. 215–238, 2006.

[10] Christopher Eagle, Mohamed Omar, Daniel Panario, and Bruce Richmond, "Distribution of the number of encryptions in revocation schemes for stateless receivers," in *Fifth Colloquium on Mathematics and Computer Science*, Uwe Roesler, Jan Spitzmann, and Marie-Christine Ceulemans, Eds. 2008, vol. AI of *DMTCS Proceedings*, pp. 195–206, Discrete Mathematics and Theoretical Computer Science.

[11] Thomas Martin, Keith M. Martin, and Peter R. Wild, "Establishing the broadcast efficiency of the subset difference revocation scheme," *Des. Codes Cryptography*, vol. 51, no. 3, pp. 315–334, 2009.

[12] Per Austrin and Gunnar Kreitz, "Lower bounds for subset cover based broadcast encryption," in *AFRICACRYPT*, Serge Vaudenay, Ed. 2008, vol. 5023 of *Lecture Notes in Computer Science*, pp. 343–356, Springer.

[13] Michael Luby and Jessica Staddon, "Combinatorial bounds for broadcast encryption," in *EUROCRYPT*, Kaisa Nyberg, Ed. 1998, vol. 1403 of *Lecture Notes in Computer Science*, pp. 512–526, Springer.

[14] Carles Padró, Ignacio Gracia, and Sebastià Martín Molleví, "Improving the trade-off between storage and communication in broadcast encryption schemes," *Discrete Applied Mathematics*, vol. 143, no. 1-3, pp. 213–220, 2004.

[15] Nuttapong Attrapadung, Kazukuni Kobara, and Hideki Imai, "Sequential key derivation patterns for broadcast encryption and key predistribution schemes," in *ASIACRYPT*, Chi-Sung Laih, Ed. 2003, vol. 2894 of *Lecture Notes in Computer Science*, pp. 374–391, Springer.

[16] Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia, "Efficient tree-based revocation in groups of low-state devices," in *CRYPTO*, Matthew K. Franklin, Ed. 2004, vol. 3152 of *Lecture Notes in Computer Science*, pp. 511–527, Springer.

[17] Carles Padró, Ignacio Gracia, Sebastià Martín Molleví, and Paz Morillo, "Linear broadcast encryption schemes," *Discrete Applied Mathematics*, vol. 128, no. 1, pp. 223–238, 2003.

[18] Carles Padró, Ignacio Gracia, Sebastià Martín Molleví, and Paz Morillo, "Linear key predistribution schemes," *Des. Codes Cryptography*, vol. 25, no. 3, pp. 281–298, 2002.

[19] Nam-Su Jho, Jung Yeon Hwang, Jung Hee Cheon, Myung-Hwan Kim, Dong Hoon Lee, and Eun Sun Yoo, "One-way chain based broadcast encryption schemes," in *EUROCRYPT*, Ronald Cramer, Ed. 2005, vol. 3494 of *Lecture Notes in Computer Science*, pp. 559–574, Springer.

[20] Yevgeniy Dodis and Nelly Fazio, “Public key trace and revoke scheme secure against adaptive chosen ciphertext attack,” in *Public Key Cryptography*, Yvo Desmedt, Ed. 2003, vol. 2567 of *Lecture Notes in Computer Science*, pp. 100–115, Springer.

[21] Dan Boneh, Craig Gentry, and Brent Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in *CRYPTO*, Victor Shoup, Ed. 2005, vol. 3621 of *Lecture Notes in Computer Science*, pp. 258–275, Springer.

Palash Sarkar Palash Sarkar received his Bachelor of Electronics and Telecommunication Engineering degree in the year 1991 from Jadavpur University, Kolkata and Master of Technology in Computer Science in the year 1993 from the Indian Statistical Institute, Kolkata. He completed his Ph.D from the Indian Statistical Institute in 1999. Since June 2005 he has been a professor at the Indian Statistical Institute. His research interests include cryptology, discrete mathematics and computer science.

Sanjay Bhattacherjee Sanjay Bhattacherjee received his Bachelor of Information Technology Engineering degree in the year 2005 from Jadavpur University, Kolkata and Master of Technology in Computer Science in the year 2009 from the Indian Statistical Institute, Kolkata. Since 2009, he is pursuing his Ph.D from the Indian Statistical Institute. His research interests include cryptology and combinatorics.

Supplementary Material

Further Results on Storage

Lemma 3. *Let $\ell_0 = d(e-1)+p$ with $1 \le p \le d$ and consider the layering strategies $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$ whose layer lengths are respectively given by $(\underbrace{d,\ldots,d}_{e-1},p)$ and $(\underbrace{d,\ldots,d}_{e-d+p},\underbrace{d-1,\ldots,d-1}_{d-p})$. Then $\mathsf{storage}_0(\boldsymbol{\ell}) = \mathsf{storage}_0(\boldsymbol{\ell}')$.*

Proof. From (1)

$$\begin{aligned}
&\mathsf{storage}_0(\boldsymbol{\ell}) - \mathsf{storage}_0(\boldsymbol{\ell}') \\
= \;& (d-p) - \frac{(d-p)(d-p+1)}{2} \\
& - \frac{d(d-1)}{2} + \frac{p(p-1)}{2} + (d-1)(d-p) \\
= \;& -\frac{(d-p)^2-(d-p)}{2} + \frac{(d-p)^2-(d-p)}{2} \\
= \;& 0.
\end{aligned}$$

□

We provide below some simple facts about storage.

1. Let $\mathbf{d} = (d_1,\ldots,d_e)$ and suppose that $d_i = d+\delta$ and $d_{e-j+1} = d$, i.e., the i-th layer length from the top is $d+\delta$ and the j-th layer length from the bottom is d. Suppose that $\mathbf{d}'$ is obtained from $\mathbf{d}$ by incrementing d_i (i.e., changing its value to $d+\delta+1$) and decrementing d_{e-j+1} (i.e., changing its value to $d-1$). Let $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$ be the corresponding sequences of special levels. A simple calculation based on (3) shows that $\mathsf{storage}_0(\boldsymbol{\ell}) - \mathsf{storage}_0(\boldsymbol{\ell}') = (e-i-j-\delta)$. So, if $e > i+j+\delta$, then it is possible to reduce storage by incrementing d_i and decrementing d_{e-j+1}. This simple observation can be used to show that the storage requirement of a layering scheme with unequal layer lengths can be reduced below a layering scheme with equal layer lengths.

   Let ℓ_0 be a positive integer and assume that d divides ℓ_0 such that $\ell_0 = d \times e$. Consider the layering scheme with layer lengths $\mathbf{d} = (d,d,\ldots,d)$. Let $\theta \ge 1$ be such that $e > 2\theta$ and define

   $$\mathbf{d}' = (\underbrace{d+1,\ldots,d+1}_{\theta},d,\ldots,d,\underbrace{d-1,\ldots,d-1}_{\theta}).$$

   Then $\mathsf{storage}_0(\boldsymbol{\ell}) = \mathsf{storage}_0(\boldsymbol{\ell}') + \theta(e-\theta-1)$. The gap $\theta(e-\theta-1)$ is positive.

2. Having a single layer of length d_e at the bottom of the tree is the same as having d_e+1 layers of length 1 each at the bottom. A simple calculation based on (3) shows this.

3. Suppose $\mathbf{d} = (d_1,\ldots,d_e)$ with $d_1 \ge d_2 \ge \cdots \ge d_e$ and $\mathbf{d}' = (d_{\pi(1)},\ldots,d_{\pi(e)})$ where π is a permutation of $\{1,\ldots,e\}$. Let $\boldsymbol{\ell}$ and $\boldsymbol{\ell}'$ be the corresponding sequences of special levels. Then $\mathsf{storage}_0(\boldsymbol{\ell}) \le \mathsf{storage}_0(\boldsymbol{\ell}')$. The quantity $\ell_0(e+1)$ and the quadratic terms in (3) are the same in both cases. A simple argument then shows the required inequality. As an example, suppose $\ell_0 = 12$ and fix $e = 8$. Then the scheme having $(d_1,d_2,\ldots,d_8) = (2,2,2,2,1,1,1,1)$ requires a storage of 50 labels whereas the scheme having $(d_1,d_2,\ldots,d_8) = (1,1,1,1,2,2,2,2)$ requires a storage of 66 labels.

Proof of Proposition 1

Proof. The bound is independent of the actual layering strategy. The upper bound of $2r-1$ for the SD scheme was already given in [3] and in [5] it was shown that this also holds for the CTSD scheme. Using the layering strategy, each subset returned by the SD algorithm can split into at most two subsets. So, if the number of SD subsets is at most $2r-1$, then there are at most $4r-2$ subsets.

Suppose the header consists of h subsets out of which h_1 are singleton sets and h_2 sets have 2 or more elements each. For each node in a singleton privileged set, its sibling (if there is one) must be a revoked user. Among all these leaves, there is only one which may not have a sibling that is also a leaf node (and this is the first privileged user from the left at level 1, for odd n). So, for the h_1 privileged users, there are at least h_1-1 other revoked users. This accounts for at least $h_1+h_1-1+2h_2=2h-1$ users. It is now easy to argue that if $h>\lceil n/2\rceil$, then $2h-1$ is greater than n. Since the total number users is n, this cannot happen. So $h\leq\lceil n/2\rceil$.

Since each subset in the subset cover will have at least one privileged user, the maximum number of subsets in the header is equal to the number of non-revoked users which is equal to $n-r$.

The bound of $4r-2$ holds for both the cases when the root is or is not a special level. If the root is a special level the bound of $4r-2$ can be improved to $4r-3$. We first provide a short argument to justify that in the SD scheme if the header length is $2r-1$, then there is a subset of the form $\mathcal{S}^0\setminus\mathcal{S}^j$ in the header. As mentioned earlier, such a subset is added to the header if and only if exactly one of the subtrees of the root node do not contain any revoked user. So, if such a subset is not in the header, then both the subtrees of the root node contain at least one revoked user. Suppose the number of revoked users in these two subtrees are r_1 and r_2 where $r=r_1+r_2$. Applying the bound on the maximum header length, we have the header to be of maximum length $2r_1-1+2r_2-1=2r-2$. So, if the header length is $2r-1$, then there must be a subset of the type $\mathcal{S}^0\setminus\mathcal{S}^j$ in the header. Using the layering strategy, each subset returned by the SD algorithm can split into at most two subsets. So, if the number of SD subsets is at most $2r-2$, then there are at most $4r-4$ subsets. On the other hand, if the number of SD subsets is equal to $2r-1$, then as argued above there must an SD subset of the form $\mathcal{S}^0\setminus\mathcal{S}^j$ in the header. Since the root node 0 is considered to be a special node, this subset will not split while all other subsets may split into two. As a result, there can be at most $4r-3$ subsets in the header. $\square$

Proof of Proposition 2

Proof. We consider the case when i is not the root and j is not a leaf. The other cases are similar. When i is not the root and j is not a leaf, the event $W_{n,r}^{i,j}=1$ is equivalent to the event $R_{sb}^{i,j}\wedge\overline{R_{rm}^{i,j}}\wedge R_{lt}^{i,j}\wedge R_{rt}^{i,j}$. We now compute as follows.

$$
\begin{aligned}
&\Pr[R^{i,j}_{sb} \wedge \overline{R^{i,j}_{rm}} \wedge R^{i,j}_{lt} \wedge R^{i,j}_{rt}] \\
&= \Pr[R^{i,j}_{sb} \wedge R^{i,j}_{lt} \wedge R^{i,j}_{rt} | \overline{R^{i,j}_{rm}}] \times \Pr[\overline{R^{i,j}_{rm}}] \\
&= \Big(1 - \Pr[\overline{R^{i,j}_{sb} \wedge R^{i,j}_{lt} \wedge R^{i,j}_{rt}} | \overline{R^{i,j}_{rm}}]\Big) \times \Pr[\overline{R^{i,j}_{rm}}] \\
&= (1 - \Pr[\overline{R^{i,j}_{sb}} | \overline{R^{i,j}_{rm}}] - \Pr[\overline{R^{i,j}_{lt}} | \overline{R^{i,j}_{rm}}] - \Pr[\overline{R^{i,j}_{rt}} | \overline{R^{i,j}_{rm}}] \\
&\qquad + \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{lt}} | \overline{R^{i,j}_{rm}}] + \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{rt}} | \overline{R^{i,j}_{rm}}] \\
&\qquad + \Pr[\overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rt}} | \overline{R^{i,j}_{rm}}] \\
&\qquad - \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rt}} | \overline{R^{i,j}_{rm}}]) \times \Pr[\overline{R^{i,j}_{rm}}] \\
&= (\Pr[\overline{R^{i,j}_{rm}}] - \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{rm}}] - \Pr[\overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rm}}] \\
&\qquad - \Pr[\overline{R^{i,j}_{rt}} \wedge \overline{R^{i,j}_{rm}}] + \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rm}}] \\
&\qquad + \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{rt}} \wedge \overline{R^{i,j}_{rm}}] + \Pr[\overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rt}} \wedge \overline{R^{i,j}_{rm}}] \\
&\qquad - \Pr[\overline{R^{i,j}_{sb}} \wedge \overline{R^{i,j}_{lt}} \wedge \overline{R^{i,j}_{rt}} \wedge \overline{R^{i,j}_{rm}}]) \\
&= \eta_r(n, \lambda_i - \lambda_j) \\
&\qquad -\eta_r(n, \lambda_s + \lambda_i - \lambda_j) \\
&\qquad -\eta_r(n, \lambda_{2j+1} + \lambda_i - \lambda_j) \\
&\qquad -\eta_r(n, \lambda_{2j+2} + \lambda_i - \lambda_j) \\
&\qquad +\eta_r(n, \lambda_s + \lambda_{2j+1} + \lambda_i - \lambda_j) \\
&\qquad +\eta_r(n, \lambda_s + \lambda_{2j+2} + \lambda_i - \lambda_j) \\
&\qquad +\eta_r(n, \lambda_{2j+1} + \lambda_{2j+2} + \lambda_i - \lambda_j) \\
&\qquad -\eta_r(n, \lambda_s + \lambda_{2j+1} + \lambda_{2j+2} + \lambda_i - \lambda_j).
\end{aligned} \tag{17}
$$

The above expression is obtained by conditioning on the event $\overline{R^{i,j}_{rm}}$ and so for the computation to go through one needs to assume that the probability of this event is positive. In the case where this probability is zero, one can directly verify that the probabilities on both sides are zero. $\square$

e \ ℓ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1(0)	3(0)	6(0)	10(0)	15(0)	21(0)	28(0)	36(0)	45(0)	55(0)	66(0)	78(0)	91(0)	105(0)	120(0)	136(0)
2		3(1)	5(1)	8(1,2)	11(2)	15(2,3)	19(3)	24(3,4)	29(4)	35(4,5)	41(5)	48(5,6)	55(6)	63(6,7)	71(7)	80(7,8)
3			6(2)	8(2)	11(2,3)	14(3)	18(3,4)	22(4,5)	26(5)	31(5,6)	36(6,7)	41(7)	47(7,8)	53(8,9)	59(9)	66(9,10)
4				10(3)	12(3)	15(3,4)	18(4)	22(4,5)	26(5,6)	30(6)	35(6,7)	40(7,8)	45(8,9)	50(9)	56(9,10)	62(10,11)
5					15(4)	17(4)	20(4,5)	23(5)	27(5,6)	31(6,7)	35(7)	40(7,8)	45(8,9)	50(9,10)	55(10)	61(10,11)
6						21(5)	23(5)	26(5,6)	29(6)	33(6,7)	37(7,8)	41(8)	46(8,9)	51(9,10)	56(10,11)	61(11)
7							28(6)	30(6)	33(6,7)	36(7)	40(7,8)	44(8,9)	48(9)	53(9,10)	58(10,11)	63(11,12)
8								36(7)	38(7)	41(7,8)	44(8)	48(8,9)	52(9,10)	56(10)	61(10,11)	66(11,12)
9									45(8)	47(8)	50(8,9)	53(9)	57(9,10)	61(10,11)	65(11)	70(11,12)
10										55(9)	57(9)	60(9,10)	63(10)	67(10,11)	71(11,12)	75(12)
11											66(10)	68(10)	71(10,11)	74(11)	78(11,12)	82(12,13)
12												78(11)	80(11)	83(11,12)	86(12)	90(12,13)
13													91(12)	93(12)	96(12,13)	99(13)
14														105(13)	107(13)	110(13,14)
15															120(14)	122(14)
16																136(15)

e \ ℓ	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	153(0)	171(0)	190(0)	210(0)	231(0)	253(0)	276(0)	300(0)	325(0)	351(0)	378(0)	406(0)	435(0)	465(0)	496(0)	528(0)
2	89(8)	99(8,9)	109(9)	120(9,10)	131(10)	143(10,11)	155(11)	168(11,12)	181(12)	195(12,13)	209(13)	224(13,14)	239(14)	255(14,15)	271(15)	288(15,16)
3	73(10,11)	80(11)	88(11,12)	96(12,13)	104(13)	113(13,14)	122(14,15)	131(15)	141(15,16)	151(16,17)	161(17)	172(17,18)	183(18,19)	194(19)	206(19,20)	218(20,21)
4	68(11,12)	74(12)	81(12,13)	88(13,14)	95(14,15)	102(15)	110(15,16)	118(16,17)	126(17,18)	134(18)	143(18,19)	152(19,20)	161(20,21)	170(21)	180(21,22)	190(22,23)
5	67(11,12)	73(12,13)	79(13,14)	85(14)	92(14,15)	99(15,16)	106(16,17)	113(17,18)	120(18)	128(18,19)	136(19,20)	144(20,21)	152(21,22)	160(22)	169(22,23)	178(23,24)
6	67(11,12)	73(12,13)	79(13,14)	85(14,15)	91(15)	98(15,16)	105(16,17)	112(17,18)	119(18,19)	126(19,20)	133(20)	141(20,21)	149(21,22)	157(22,23)	165(23,24)	173(24,25)
7	68(12)	74(12,13)	80(13,14)	86(14,15)	92(15,16)	98(16)	105(16,17)	112(17,18)	119(18,19)	126(19,20)	133(20,21)	140(21)	148(21,22)	156(22,23)	164(23,24)	172(24,25)
8	71(12,13)	76(13)	82(13,14)	88(14,15)	94(15,16)	100(16,17)	106(17)	113(17,18)	120(18,19)	127(19,20)	134(20,21)	141(21,22)	148(22)	156(22,23)	164(23,24)	172(24,25)
9	75(12,13)	80(13,14)	85(14)	91(14,15)	97(15,16)	103(16,17)	109(17,18)	115(18)	122(18,19)	129(19,20)	136(20,21)	143(21,22)	150(22,23)	157(23)	165(23,24)	173(24,25)
10	80(12,13)	85(13,14)	90(14,15)	95(15)	101(15,16)	107(16,17)	113(17,18)	119(18,19)	125(19)	132(19,20)	139(20,21)	146(21,22)	153(22,23)	160(23,24)	167(24)	175(24,25)
11	86(13)	91(13,14)	96(14,15)	101(15,16)	106(16)	112(16,17)	118(17,18)	124(18,19)	130(19,20)	136(20)	143(20,21)	150(21,22)	157(22,23)	164(23,24)	171(24,25)	178(25)
12	94(13,14)	98(14)	103(14,15)	108(15,16)	113(16,17)	118(17)	124(17,18)	130(18,19)	136(19,20)	142(20,21)	148(21)	155(21,22)	162(22,23)	169(23,24)	176(24,25)	183(25,26)
13	103(13,14)	107(14,15)	111(15)	116(15,16)	121(16,17)	126(17,18)	131(18)	137(18,19)	143(19,20)	149(20,21)	155(21,22)	161(22)	168(22,23)	175(23,24)	182(24,25)	189(25,26)
14	113(14)	117(14,15)	121(15,16)	125(16)	130(16,17)	135(17,18)	140(18,19)	145(19)	151(19,20)	157(20,21)	163(21,22)	169(22,23)	175(23)	182(23,24)	189(24,25)	196(25,26)
15	125(14,15)	128(15)	132(15,16)	136(16,17)	140(17)	145(17,18)	150(18,19)	155(19,20)	160(20)	166(20,21)	172(21,22)	178(22,23)	184(23,24)	190(24)	197(24,25)	204(25,26)
16	138(15)	141(15,16)	144(16)	148(16,17)	152(17,18)	156(18)	161(18,19)	166(19,20)	171(20,21)	176(21)	182(21,22)	188(22,23)	194(23,24)	200(24,25)	206(25)	213(25,26)
17	153(16)	155(16)	158(16,17)	161(17)	165(17,18)	169(18,19)	173(19)	178(19,20)	183(20,21)	188(21,22)	193(22)	199(22,23)	205(23,24)	211(24,25)	217(25,26)	223(26)
18		171(17)	173(17)	176(17,18)	179(18)	183(18,19)	187(19,20)	191(20)	196(20,21)	201(21,22)	206(22,23)	211(23)	217(23,24)	223(24,25)	229(25,26)	235(26,27)
19			190(18)	192(18)	195(18,19)	198(19)	202(19,20)	206(20,21)	210(21)	215(21,22)	220(22,23)	225(23,24)	230(24)	236(24,25)	242(25,26)	248(26,27)
20				210(19)	212(19)	215(19,20)	218(20)	222(20,21)	226(21,22)	230(22)	235(22,23)	240(23,24)	245(24,25)	250(25)	256(25,26)	262(26,27)
21					231(20)	233(20)	236(20,21)	239(21)	243(21,22)	247(22,23)	251(23)	256(23,24)	261(24,25)	266(25,26)	271(26)	277(26,27)
22						253(21)	255(21)	258(21,22)	261(22)	265(22,23)	269(23,24)	273(24)	278(24,25)	283(25,26)	288(26,27)	293(27)
23							276(22)	278(22)	281(22,23)	284(23)	288(23,24)	292(24,25)	296(25)	301(25,26)	306(26,27)	311(27,28)
24								300(23)	302(23)	305(23,24)	308(24)	312(24,25)	316(25,26)	320(26)	325(26,27)	330(27,28)
25									325(24)	327(24)	330(24,25)	333(25)	337(25,26)	341(26,27)	345(27)	350(27,28)
26										351(25)	353(25)	356(25,26)	359(26)	363(26,27)	367(27,28)	371(28)
27											378(26)	380(26)	383(26,27)	386(27)	390(27,28)	394(28,29)
28												406(27)	408(27)	411(27,28)	414(28)	418(28,29)
29													435(28)	437(28)	440(28,29)	443(29)
30														465(29)	467(29)	470(29,30)
31															496(30)	498(30)
32																528(31)

Table 1: $\#\mathrm{SML}_0(e,\ell_0)$ and $\Lambda(e,\ell_0)$ for $1 \le \ell_0 \le 32$ and $1 \le e \le \ell_0$.