Complete Tree Subset Difference Broadcast Encryption Scheme and its Analysis

Sanjay Bhattacherjee
Applied Statistics Unit
Indian Statistical Institute
203, B.T.Road, Kolkata, India - 700108.
sanjayb_r@isical.ac.in

Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute
203, B.T.Road, Kolkata, India - 700108.
palash@isical.ac.in

Abstract

The Subset Difference (SD) method proposed by Naor-Naor-Lotspeich is the most popular broadcast encryption (BE) scheme. It is suitable for real-time applications like Pay-TV. It has been suggested for use by the AACS standard for digital rights management in Blu-Ray and DVD discs. The SD method assumes the number of users to be a power of two. (1) We propose the Complete Tree Subset Difference (CSD) method that subsumes the SD method by allowing arbitrary number of users in the system. All the results obtained in this work for the CSD scheme hold good for the SD scheme by assuming the number of users to be the next power of two. (2) Given the importance of the SD scheme, its detailed combinatorial analysis is of practical interest. We find recurrences for the CSD scheme to count the number of possible ways r users in the system of n users can be revoked to result in a transmission overhead (header length) of h. The header length h of a broadcast is an important efficiency parameter in BE. The usefulness of these recurrences is demonstrated by generating exhaustive data of the above count, obtaining bounds on the header length and various other interesting results some of which are difficult to prove without the recurrences. (3) An $O(r \log n)$ time algorithm is proposed to compute the expected header length in the CSD scheme for n users in the system, r out of which are revoked. This algorithm is of practical interest in its own right, for efficiency and performance analysis of the CSD scheme. Using this algorithm, we show that for practical values of n and r, the transmission efficiency of the CSD scheme is better than the SD scheme. For n a power of two and a fixed $r \geq 2$, we obtain an upper bound on the expected header length and show that this bound is also the limit as $n \to \infty$.

1 Introduction

1.1 What is Broadcast Encryption?

The cryptographic method for a centre to efficiently broadcast encrypted digital content to a system of users so that only an intended subset (the *privileged* users) can correctly decrypt it is called *Broadcast Encryption* (BE). Before the system starts to work, the users are given some secret information (may be the secret keys or some information from which it can derive the secret keys). A user uses this information for decrypting the encrypted digital content intended for itself.

In a typical BE scheme, each message (a block of digital content) that is broadcast is encrypted using a unique key called a *session key*. The session key in turn, is encrypted a number of times using user keys and these multiple encryptions of the session key is sent as the *header* of the encrypted message. The transmission overhead of the scheme is determined by the *header length* h (the number of encryptions of the session key in the header).

In a fully resilient scheme, even if an adversary has the decryption keys of all the remaining non-privileged users in the system (the *revoked* users), it will not be able to correctly decrypt the content. A crucial requirement

for a BE scheme is that it should facilitate dynamic revocation of decryption privilege from any subset of users at any point of time (based on their subscription or privilege status).

Importance of Broadcast Encryption: Copyright protection using Digital Rights Management [DRM] techniques is an important application of BE. Out of the different facets of copyright protection, BE handles the content protection part. The application of BE systems is pretty wide in the implementation of [DRM] for content protection in digital data distribution technologies such as pay-TV, Internet or mobile video broadcast, optical discs, etcetera.

Requirements from a BE scheme: In real-time scenarios like Pay-TV, Internet or mobile video broadcast, the number of users can vary from a few thousands to millions. For other real-time applications of BE like broadcasting secret instructions to military outposts from a base station, the number of users will be much smaller (maximum of a few hundreds). The BE scheme that is used in real time scenarios as above, has to be efficient in terms of the transmission overhead associated with each message as also the encryption and decryption times and storage of user keys.

For non-real-time applications like content protection in Blu-Ray discs and DVDs (optical discs), the requirements from a BE scheme are somewhat different. The number of users (disc players) in such scenarios maybe in millions. The transmission overhead (the additional information stored in the physical media, that is used for decrypting the content) is not really an issue since storage space in discs is no more a constraint nowadays. Further, since encryption does not happen in real-time, improving the encryption time is also not very important. On the other hand, reducing the user storage (number of keys or their equivalent secret to be stored in the player) and decryption time is still important.

Importance of the [NNL01] SD scheme: Broadcast Encryption was introduced in [Ber91] followed by [FN93]. There have been several works in this area [Sti97], [SW98] since then, but the most popular scheme out of these is the tree-based Subset Difference (SD) method of [NNL01]. Since it is a symmetric key based scheme, it is very efficient in terms of encryption and decryption time. It allows the users to be stateless (users do not have to update their individual secret information with every session) and also allows dynamic revocation of users. User storage requirement is $O(\log^2 n)$ where n is the total number of users and the transmission overhead is linear in the number of revoked users r. Currently, the SD scheme offers the simplest algorithm and the best trade-offs for use in both real-time applications like Pay-TV and non-real time applications like content protection in optical discs [AAC].

1.2 Our Contributions:

Arbitrary number of users: In this paper, we broaden the scope of use of the SD scheme. The SD scheme and all follow-up works [HS02, PB06, AK08, MMW09] assume the total number of users n to be a power of two. We relax this restriction to allow any arbitrary number of users in the system by introducing the Complete Tree Subset Difference (CSD) scheme. The CSD scheme is based on the SD scheme and subsumes it. When the number of users in the CSD method is a power of two, it becomes exactly the same as the SD scheme.

When implementing the SD scheme for applications such as Pay-TV, it is most likely that the number of users in the system will not be a power of two. In that case the centre has to assume the existence of dummy users to make the number of users a power of two. The CSD scheme on the other hand, can accommodate an arbitrary number of users, thus eliminating the requirement of dummy users in the system.

Inclusion of dummy users results in the expected header length of the SD scheme to be more than the CSD scheme for practical values of n and r. This is intuitive and we provide further arguments and supporting data in Section 3.2 and Section 5.4. In real-time scenarios like Pay-TV, where many messages are transmitted, avoiding the extra bandwidth arising due to dummy users will be desirable. Consequently, one would prefer to exclude them and instead work with the actual number of users present in the system.

It is to be noted that an implementation that uses the SD scheme, can easily shift to using the CSD scheme with minimal change in the software implementation. This is because the internal tree structure used for assigning keys to subsets of users in the SD scheme remains almost the same in the CSD scheme.

Combinatorial Analysis: The importance of the (C)SD scheme motivates the study of its combinatorial properties. We use a new approach for a detailed combinatorial study of the CSD scheme. A method is proposed to count the number of ways that r out of n users can be revoked to get a header length of h in the CSD scheme. This counting is formulated using two recurrences. Since the SD scheme is a special case of the CSD scheme, these recurrences hold for the SD scheme too.

Using these recurrences, a dynamic programming based algorithm to do the above counting is developed. Previous to this work, the only known method to do such counting was to run the SD algorithm itself on all possible $\binom{n}{r}$ revocation patterns. The resulting time complexity can be exponential in n. In contrast, the running time of the algorithm proposed here is always polynomial in n. This, by itself, is a significant improvement.

The importance of these recurrences in capturing the detailed combinatorial properties of the CSD scheme is demonstrated by obtaining important results from them.

1. The worst case header length for a given r in the SD scheme was shown to be $2r-1$ in [NNL01]. We show that the worst case header length for the CSD scheme is $\min(2r-1, \lceil n/2 \rceil, n-r)$.

2. Given r, we characterize the minimum number of users n_r (that need to be in a system using the CSD method), that can give rise to the maximum header length of $2r-1$.

3. For the special case when n is a power of two (i.e., for the SD scheme), we use the recurrences to obtain a generating function for the sequence. Earlier, a generating function of a slightly different form was obtained in [PB06] using direct arguments. We did not attempt to find the generating function for the case when n is not a power of two. This would be quite cumbersome and did not appear to be of much interest.

Probabilistic analysis: We propose a new and efficient $O(r \log n)$ algorithm for computing the expected header length in the CSD method for a given n and r. This algorithm is based on the probabilistic analysis of revocation of users. It is simple to implement. The crucial importance of the algorithm lies in the fact that it enables us to explore in depth the behaviour of the expected header length for values of n ranging from a few hundreds to a million. Examples of outputs obtained by running the algorithm are provided later. We believe that the algorithm for computing the expected header length will be a very useful tool for practitioners implementing the (C)SD scheme.

When n is a power of two (i.e., the SD scheme), we show that the expected header length for r revoked users is bounded above by

$$(3r-2) - 3 \times \sum_{i=1}^{r-1} \left(\left(-\frac{1}{2}\right)^i + \sum_{k=1}^{i} (-1)^k \binom{i}{k} \frac{(2^k - 3^k)}{(2^k - 1)} \right).$$

We further show that as $n \to \infty$ through powers of two, this is actually the limiting value of the expected header length. Computing the above expression for different values of r shows it to be always less than $1.25r$.

The previously known upper bound on the expected header length in the SD scheme for r revoked users was proved to be $1.38r$ in [NNL01]. They commented that experimental results indicated that the bound is probably

$1.25r$. Our analysis of the expected header length shows the precise limiting upper bound and clarifies the issue of this value being $1.25r$.

1.3 Previous and related works:

The tree-based SD scheme has inspired quite a lot of work in the area of broadcast encryption. Asymptotic improvements to the user storage parameter of the SD scheme were suggested in the tree-based LSD scheme of [HS02] with some loss of efficiency in the transmission overhead. Analysis of the combinatorics behind broadcast encryption schemes and different generic bounds on the efficiency parameters have been done in [LS98, PGM04] and other works. A generic method for constructing BE schemes from pseudo-random generators was proposed in [AKI03].

An analysis of the expected header length of the SD and LSD schemes was done in [PB06]. As mentioned earlier, they proposed generating functions for counting the number of ways p users (out of total n users) can be given access privilege so that the header length will be h. Using this generating function, they found equations to compute the expected header length for a given n and r. However, they admitted that their equations were "complex to compute and difficult to gain insight from". Consequently they went forward to find *approximations* for the same. The analysis of the expected header length in [PB06] was continued in [EOPR] to show that the standard deviations are small compared to the means as the number of users gets large. Other combinatorial studies of the SD method has been done in [MMW09, AK08]. In particular, the maximum possible header length for a given n and r was found accurately in [MMW09].

Extension of [BS11]: This work is the extended and considerably modified version of [BS11]. The work in [BS11] was the first to propose accommodating arbitrary number of users by modifying the SD method (that uses a *full tree*) to use an *incomplete tree* with the users as its leaves. In the current work, considerable changes have been made in the structure of the tree underlying the scheme. This constitutes the main difference between the current work and that in [BS11]. Although the capability of accommodating arbitrary number of users has been retained, a *balanced complete tree* structure has been used for assignment of keys to subsets (in place of the unbalanced incomplete tree used in [BS11]). Recurrences to analyze the CSD method has been found in a manner similar to the one found in [BS11]. The algorithm to compute the expected header length in [BS11] has been modified to obtain a similar algorithm for the CSD method of this paper. In Appendix A, we very briefly describe the results of [BS11].

Other related work: There are several other BE schemes. A family of broadcast encryption schemes using linear algebraic techniques and hence called linear broadcast encryption schemes was introduced in [PGMM03]. The same authors had also proposed key pre-distribution techniques based on linear algebraic techniques in [PGMM02]. Another interesting work on BE is [JHC+05]. It works on the idea of "one key per punctured interval" in which the worst case header length has been brought down to r (or below at the cost of increasing user storage) for the first time. But, the method is more complicated than the SD scheme and the user storage requirement is rather high.

Traitor tracing is a related issue. We do not discuss this here, since it is not connected to the contribution of the paper. We only remark that the traitor tracing method for the SD scheme can be modified to obtain a traitor tracing method for the CSD scheme. There are several schemes on public-key BE which we do not consider at all.

2 The Subset Cover Revocation Framework

The Complete Tree Subset Difference method that we propose is based on the Subset Difference method introduced by Naor Naor Lotspeich in [NNL01]. The Subset Difference algorithm is essentially a key encrypting

method that falls under the Subset Cover Revocation Framework that was proposed in the same paper. We begin with a very short description of this framework.

The Subset Cover Revocation Framework assumes a *centre* that encrypts a message M and broadcasts it to a set $\mathcal{N}$ of ($|\mathcal{N}| =$) n *users*. This set of users are all the possible recipients of the broadcast. A subset $\mathcal{R}$ ($\subseteq \mathcal{N}$) of these users are revoked (say non-subscribers of a service). The centre broadcasts using a broadcast encryption algorithm such that any user belonging to the set $\mathcal{N} \setminus \mathcal{R}$ should be able to correctly decrypt the message M from the broadcast, while any coalition of users belonging to the set $\mathcal{R}$ should not be able to correctly decrypt it.

A broadcast encryption algorithm under this framework consists of three parts: (1) *an initiation scheme* - that assigns user $u \in \mathcal{N}$ secret information I_u that will allow them to decrypt messages intended for them; (2) *the broadcast algorithm* - that takes as input the message M and the set $\mathcal{R}$ of revoked users and outputs the ciphertext C. C is broadcast to all the users in $\mathcal{N}$; (3) *the decryption algorithm* - that runs at the user end. It takes as input the ciphertext C and the secret information I_u that the user u had received during initiation and attempts to decrypt C. A privileged user should be able to get back the original message M, while a revoked user should not be able to get back the correct message from C.

During initiation, an algorithm in the framework defines a collection $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_w\}$ of subsets, where each $\mathcal{S}_j \subseteq \mathcal{N}$. Each subset $\mathcal{S}_j$ is assigned a long-lived key L_j. This assignment may not be explicit as we will see in the Complete Tree Subset Difference algorithm. However, a user $u \in \mathcal{S}_j$ should be able to deduce L_j from the secret information I_u it had acquired during initiation. During broadcast, given a set $\mathcal{R}$ of revoked users, the set of privileged users $\mathcal{N} \setminus \mathcal{R}$ is partitioned into pairwise disjoint subsets $\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_h}$ taken from the collection $\mathcal{S}$. This partition is called the *subset cover* $\mathcal{S}_c$. In other words,

$$\mathcal{N} \setminus \mathcal{R} = \bigcup_{j=1}^{h} \mathcal{S}_{i_j}$$

where each $\mathcal{S}_{i_j} \in \mathcal{S}$ and $\mathcal{S}_c = \{\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_h}\}$. The size h of the subset cover is called the header length (we will soon see why).

During broadcast, an algorithm in the framework uses two encryption schemes:

- A function $F_K : \{0,1\}^* \to \{0,1\}^*$ to encrypt the message M with a *session key* K. The session key is a random string chosen afresh for each new message M.
- A function $E_{L_j} : \{0,1\}^* \to \{0,1\}^*$ to encrypt the session key K with a long-lived key L_j corresponding to the subset $\mathcal{S}_j$ ($\in \mathcal{S}_c$) of users.

Detailed discussion on the security requirement of these primitives can be found in [NNL01]. Hence, in order to broadcast the message M, the centre chooses a session key K and encrypts M as $F_K(M)$. The centre knows the set $\mathcal{N} \setminus \mathcal{R}$ of privileged users. It finds the cover $\mathcal{S}_c = \{\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_h}\}$. Let $L_{i_1}, \ldots, L_{i_h}$ be the long-lived keys that were assigned to each of these subsets in $\mathcal{S}_c$. The centre then encrypts the session key K with each of these keys L_{i_j}. The session key has to be encrypted h times for each set in $\mathcal{S}_c$. The h encryptions of the session key is sent along with $F_K(M)$ as a *header* for the encrypted message. The header also has information to identify the subsets $\mathcal{S}_{i_j}$ that form the cover $\mathcal{S}_c$. The size h of the header is determined by the number of sets in $\mathcal{S}_c$. We are going to refer to this size as the *header length*. The encrypted message $F_K(M)$ along with the header forms the ciphertext C.

During decryption, a user u has to identify from the header, the set $\mathcal{S}_{i_j}$ to which it belongs. It decrypts the session key K from the portion of the header that has K encrypted for $\mathcal{S}_{i_j}$ using the long-lived key L_{i_j} that it derives from the secret information I_u it had acquired during initiation. Using K, it can decrypt the message M from $F_K(M)$. In case the user does not belong to any of the sets in $\mathcal{S}_c$ (it is a revoked user), it will not be able to decrypt K or M for that matter.

3 The Complete Tree Subset Difference Method

The Subset Difference (SD) method of [NNL01] and all follow-up work assumes the number of users n to be a power of two. We propose the Complete Tree Subset Difference (CSD) algorithm that can accommodate any arbitrary number of users.

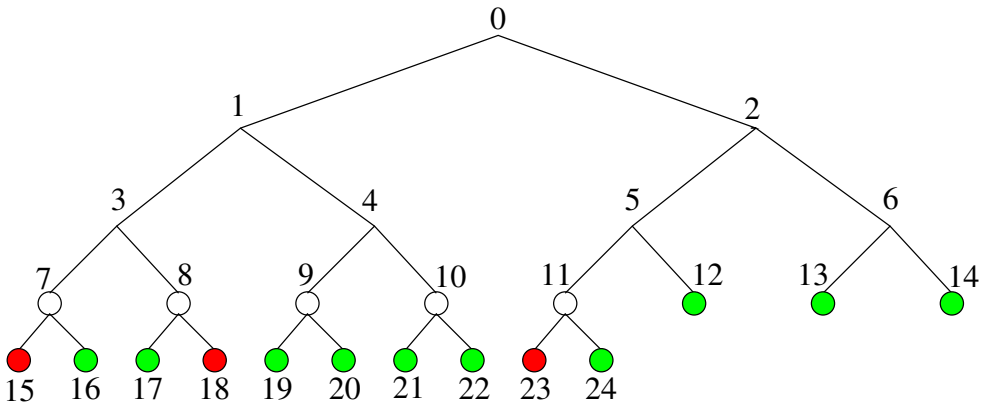

Figure 1: The *complete (non-full)* tree $\mathcal{T}^0$ with $n = 13$ users as its leaves. Privileged users are indicated in green and the revoked users are indicated in red. Here, $r = 3$. The tree $\mathcal{T}^1$ is a subtree of $\mathcal{T}^0$ and is a *full* subtree having 8 leaf nodes whereas the tree $\mathcal{T}^2$ is a non-full *complete* subtree of $\mathcal{T}^0$ with 5 leaf nodes.

Our algorithm considers a rooted complete binary tree $\mathcal{T}^0$ with n leaves. (One may note here that a *complete binary tree* has leaf nodes only at the bottom-most (last) level and maybe also the last-but-one level. The leaves in the last level are filled from the left to the right in the tree. In a *full binary tree* of height ℓ there are 2^ℓ leaves, all at the last level. A full binary tree is also complete by definition. We will refer to trees that are complete but not full as *non-full*.) Each user in $\mathcal{N}$ is associated with a leaf of the complete binary tree $\mathcal{T}^0$. There are $2n-1$ nodes (internal and leaf) in $\mathcal{T}^0$. These nodes are labeled with numbers from $\{0, 1, \ldots, 2n-2\}$. The root node of $\mathcal{T}^0$ is labeled as 0. All subsequent nodes are labeled as follows: the left child node of a node i is labeled as $2i+1$ and the right child is labeled as $2i+2$. Hence, the nodes 0 to $n-2$ are the internal nodes. The nodes labeled $n-1$ to $2n-2$ are the leaf nodes. The subtree of $\mathcal{T}^0$ rooted at node i is denoted by $\mathcal{T}^i$. The number of leaf nodes in the subtree $\mathcal{T}^i$ is denoted by λ_i.

Now that we have attached the users to the tree $\mathcal{T}^0$, we need to define the collection $\mathcal{S}$ of subsets. We define $S_{i,j}$ as the set of users in the subtree $\mathcal{T}^i$ but *not* in $\mathcal{T}^j$. This set $S_{i,j}$ is also denoted as $\mathcal{T}^i \setminus \mathcal{T}^j$. All subsets of users of the form $S_{i,j}$ where node j is a *descendant* of node i (a node in the subtree $\mathcal{T}^i$) is included in the collection $\mathcal{S}$. The set $\mathcal{N}$ of all users is also included in $\mathcal{S}$. Once this collection $\mathcal{S}$ has been created, each set $S_{i,j}$ in $\mathcal{S}$ has to be assigned a long-lived key $L_{i,j}$. We will look at the key assignment later.

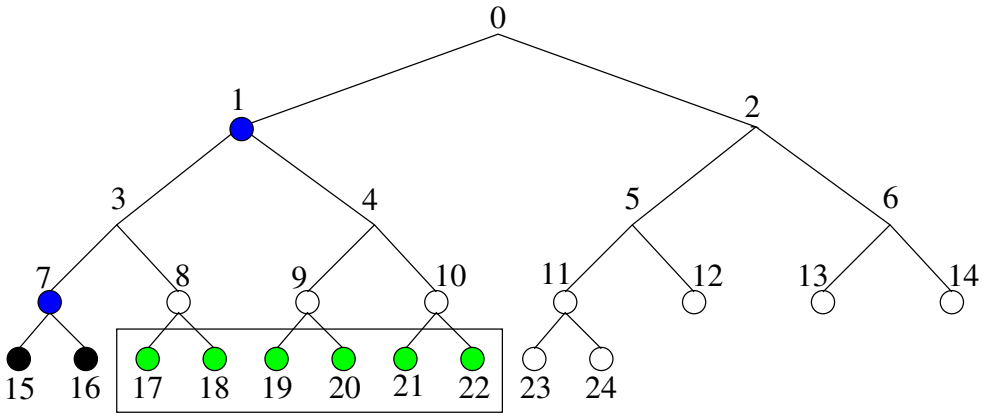

Figure 2: The *subset difference* subset $S_{1,7}$ which includes leaves in $\mathcal{T}^1$ but not in $\mathcal{T}^7$ i.e.; $S_{1,7} = \mathcal{T}^1 \setminus \mathcal{T}^7 = \{17, 18, 19, 20, 21, 22\}$.

During broadcast, the centre will know the set $\mathcal{R}$ of revoked users and the message M to be broadcast. It has to find the subset cover $\mathcal{S}_c$ for $\mathcal{N} \setminus \mathcal{R}$. $\mathcal{S}_c$ is a collection of pairwise disjoint sets $S_{i_1,j_1}, \ldots, S_{i_h,j_h}$ (each S_{i_k,j_k} taken from $\mathcal{S}$) such that $\mathcal{S}_c = \bigcup_{k=1}^{h} \mathcal{S}_{i_k,j_k}$. If there are no revoked users ($\mathcal{R}$ is empty), then the only set in the cover $\mathcal{S}_c$ is $\mathcal{N}$. Otherwise, the following *cover-finding algorithm* is used: The centre first constructs the Steiner Tree $\mathcal{ST}(\mathcal{R})$ induced by $\mathcal{R}$ on $\mathcal{T}^0$. (The Steiner Tree $\mathcal{ST}(\mathcal{R})$ is a subgraph of $\mathcal{T}^0$ that only retains the nodes and edges on paths from the root node 0 to a revoked leaf node. All the other paths in $\mathcal{T}^0$ are deleted.) The cover-finding algorithm runs iteratively by maintaining a tree $\mathcal{T}$ that is a sub-graph of $\mathcal{ST}(\mathcal{R})$. It starts by initializing $\mathcal{T}$ as a copy of $\mathcal{ST}(\mathcal{R})$. At every iteration, the algorithm keeps removing nodes from $\mathcal{T}$ (while adding subsets to $\mathcal{S}_c$) until $\mathcal{T}$ has just one node left. At any point of time in the algorithm, a leaf node in $\mathcal{T}$ corresponds to either a leaf node in $\mathcal{T}^0$ or the root of a subtree in $\mathcal{T}^0$ all whose leaves have already been covered till that iteration. More precisely:

1. If there is only one leaf node in $\mathcal{T}$, jump to step 6.
2. Find two leaves j_1 and j_2 of $\mathcal{T}$ whose common *ancestor* i (both j_1 and j_2 belong to the minimal subtree $\mathcal{T}^i$) does not have any other leaf node in its subtree in $\mathcal{T}$. (Here, out of the many possible such pairs j_1 and j_2 one may choose the leftmost to have a specific algorithm. Any other choice would have worked equally well.)
3. Let i_1 (respectively i_2) be the immediate child node of i which is an ancestor of j_1 (respectively j_2) or is the node j_1 (respectively j_2) itself. If $i_1 \neq j_1$ then add the set S_{i_1,j_1} to the cover $\mathcal{S}_c$. Similarly, if $i_2 \neq j_2$ then add the set S_{i_2,j_2} to the cover $\mathcal{S}_c$.
4. Delete the paths joining j_1 and j_2 with their common ancestor i.
5. If there are more than one leaf remaining in $\mathcal{T}$, go back to step 2.
6. If the only leaf node is the node 0 (node corresponding to the root of $\mathcal{T}^0$), then there are no more subsets to be added to $\mathcal{S}_c$. Else, add the set $S_{0,j}$ (where j is the leaf node remaining in $\mathcal{T}$) to $\mathcal{S}_c$.

3.1 Key assignment to each subset $S_{i,j}$ in $\mathcal{S}$

Pseudo-random sequence generator G: In order to assign keys to each subset in $\mathcal{S}$, the centre assigns uniform random seeds to every non-leaf node in $\mathcal{T}^0$ and uses a *cryptographic pseudo-random sequence generator*

G. The pseudo-random generator G outputs a pseudo-random string that has three times the length of the input seed. The output string $G(seed)$ is divided into three equal parts $G_L(seed)$, $G_M(seed)$ and $G_R(seed)$. (Hence, $G(seed) = G_L(seed) \parallel G_M(seed) \parallel G_R(seed)$.) $G : \{0,1\}^k \rightarrow \{0,1\}^{3k}$ is a pseudo-random sequence generator if no polynomial time adversary can distinguish between its output for a random seed with a truly random string of the same length.

Seed assignment to nodes: Every non-leaf node i in $\mathcal{T}^0$ is assigned a uniform random seed $LABEL_i$. Every non-root node j of $\mathcal{T}^0$ is assigned derived seeds from every ancestor i of j. The left child $2i+1$ of node i in $\mathcal{T}^0$ derives the seed $G_L(LABEL_i)$ from the random seed $LABEL_i$ of i. All descendants of $2i+1$ further get derived seeds from this derived seed $G_L(LABEL_i)$ of $2i+1$. Similarly, the right child $2i+2$ of node i in $\mathcal{T}^0$ derives the seed $G_R(LABEL_i)$ from the random seed of i and all descendants of $2i+2$ get derived seeds from this derived seed $G_R(LABEL_i)$ of $2i+2$. We denote the seed for a node j (that is a descendant of i) derived from the random seed of node i as $LABEL_{i,j}$. Following such an assignment of random and derived seeds for nodes in $\mathcal{T}^0$, the long lived key $L_{i,j}$ assigned to the set $S_{i,j}$ is $G_M(LABEL_{i,j})$.

I_u for each $u \in \mathcal{N}$: Once the centre is done with the assignment of seeds (uniform random as well as derived) to nodes, it has to distribute the secret information I_u to each user $u \in \mathcal{N}$. The user associated with a leaf j of $\mathcal{T}^0$ must have been revoked when a set $S_{i,j}$ is in the cover $\mathcal{S}_c$. Hence, the user at leaf j should not be able to compute the $L_{i,j}$ for any of its predecessor i in $\mathcal{T}^0$. In fact, it should not be able to compute any $L_{i,k}$ where k is also one of its ancestors (k must be a descendant of i though). In other words, a user at leaf j should be able to compute an $L_{i,k}$ if and only if i is an ancestor of j but k is not (k is not on the path joining the leaf j with i). In a subtree $\mathcal{T}^i$ of $\mathcal{T}^0$ to which a user at leaf j belongs, the node i has a random seed $LABEL_i$. The user at j gets the seeds of all nodes adjacent to the path joining i and j that have been derived from $LABEL_i$. Say $i_1, \ldots, i_m$ are those nodes "falling off" from the path between node i and leaf j. The user at j will get the derived seeds $LABEL_{i,i_1}, LABEL_{i,i_2}, \ldots, LABEL_{i,i_m}$. To summarize, the I_u for a user u at leaf j consists of all derived seeds $LABEL_{i,k}$ such that i is a predecessor of j and k is adjacent to the path joining i and j. As derived in [NNL01], the number of derived seeds in I_u is $\frac{1}{2}\log^2 n + \frac{1}{2}\log n + 1$ for n a power of two. For an arbitrary n, one has to consider the next higher power of two, say $2^{\ell_0 - 1} < n \leq 2^{\ell_0}$. The number of derived seeds in I_u will be $\frac{1}{2}\ell_0^2 + \frac{1}{2}\ell_0 + 1$.

3.2 Avoiding Dummy Users

The CSD scheme works with the actual number of users that are present in the system. It may be argued that even if n is not a power of two, the SD scheme can be applied by incorporating dummy users to make the total number of users to be a power of two. We argue that this impacts the size of the transmission overhead. For an actual broadcast, there are two ways to handle the dummy users – either consider all of them to be revoked or consider all of them to be privileged.

Suppose that the dummy users are considered to be distributed randomly among all the users. Then viewing them as revoked has very serious performance penalties. This is because, the average header length is linear in the number of revoked users (to be proved later). Having a larger number of *randomly distributed* revoked users leads to larger header size. If, on the other hand, the dummy users are viewed as privileged, then the performance penalty will be lesser. Examples for this situation is provided in Section 5.4 after developing the algorithm for computing the expected header length.

Assuming the dummy users to be randomly distributed may not be justifiable. In an actual implementation, they may be considered to be one block. Suppose that $2^{\ell-1} < n < 2^{\ell}$ and that the users numbered $n+1, \ldots, 2^{\ell}$ are the dummy users and the real users are numbered 1 to n. The actual revoked users will be among the values 1 to n, whereas the users numbered $n+1, \ldots, 2^{\ell}$ will be considered to be either all revoked or all privileged.

Consider a particular revocation pattern and a subset cover that corresponds to this revocation pattern. If there are no dummy users in the system (CSD scheme), then the subset cover must cover all the actual privileged users. Suppose now that there are dummy users all of which are privileged. Since these occur at the end, the earlier subset cover will still be required. The cover generation algorithm may introduce a few additional subsets to cover the dummy privileged users, but, there is no way that a subset from the original cover will be dropped. Similarly, if the dummy users are considered to be revoked, then the corresponding subset cover must still cover the actual privileged users and hence will contain all the subsets of the original subset cover. So, in either case, i.e., whether the dummy users are privileged or revoked, the size of the subset cover cannot be smaller than the size of the original subset cover.

We ran a few experiments to verify the above argument. The value of n is chosen to vary from 17 to 24 and the value of r is chosen to vary from 2 to 8. Table 1 shows the expected header lengths obtained by the CSD algorithm. For each value of n, we considered two possible ways of handling dummy users – all of them privileged and all of them revoked. The results for dummy revoked users is shown in Table 2 and the results for dummy privileged users is shown in Table 3.

In all cases, we observe that the CSD method is never inferior to the SD method with dummy users. In certain cases, the drop in the expected header length of the CSD method as compared to SD method with dummy users can be more than 0.5. While this may not seem very impressive, for actual practical situations, the number of users will be much more and the corresponding improvements will become noticeable. Later we use our algorithm for computing expected header lengths to report results for the CSD scheme for large values of n. But, there is no corresponding algorithm for the SD scheme with dummy revoked or dummy privileged users with all the dummy users forming a block. One needs to run the SD cover generation algorithm on all possible revocation patterns. It is not possible to do this for large values of n.

4 Counting Revocation Patterns in the CSD method

A given set of revoked users is called a *revocation pattern*. We denote a revocation pattern on n users where r are revoked, as an (n, r)-*revocation pattern*. The number of possible (n, r)-revocation patterns is $\binom{n}{r}$. In order to study the behaviour of the CSD algorithm, we find a method to count the number of (n, r)-revocation patterns that result in a cover size (header length) of h.

In a subtree $\mathcal{T}^j$ of $\mathcal{T}^0$ with λ_j users (leaves), $N(\lambda_j, r, h)$ *is defined as the number of* (λ_j, r)*-revocation patterns that are covered by exactly* h *subsets*. Similarly, for λ_j users in $\mathcal{T}^j$, $T(\lambda_j, r, h)$ *is defined as the number of* (λ_j, r)*-revocation patterns that are covered by* h *subsets such that there is at least one revoked user in both subtrees of* $\mathcal{T}^j$. Since the tree $\mathcal{T}^0$ has n $(= \lambda_0)$ leaves, $N(n, r, h) = N(\lambda_0, r, h)$ is the number of (n, r)-revocation patterns covered by a header length of h. $N(n, r, h)$ is what we intend to find.

4.1 Few notations

Level number and position of nodes: Before we start computing the values of $T(n, r, h)$ and $N(n, r, h)$, we fix a few notation for the ease of description. A level number of $\mathcal{T}^0$ is indicated by ℓ. At times we will denote the level of a node i by ℓ_i. The root node 0 is at the highest level ℓ_0. Hence, $\ell \in \{0, \ldots, \ell_0\}$. Since every subtree $\mathcal{T}^i$ is a complete binary tree, $2^{\ell_i - 1} < \lambda_i \leq 2^{\ell_i}$. For the whole tree $\mathcal{T}^0$, we see that $2^{\ell_0 - 1} < n \leq 2^{\ell_0}$. In other words, $2^{\ell_1} < n \leq 2^{\ell_1 + 1}$ where ℓ_1 is the level of node 1 (left child of root node). The number of nodes at level ℓ of $\mathcal{T}^0$ is denoted by q_ℓ. We see that the number of nodes at the last level is $q_0 = 2(n - 2^{\ell_1})$. For $\ell \in \{1, \ldots, \ell_0\}$, $q_\ell = 2^{\ell_0 - \ell}$. The position of a node at a level from the left is denoted by k where k ranges from 1 to q_ℓ. Hence, a node i is uniquely represented by the pair (ℓ_i, k_i) – the level ℓ_i of $\mathcal{T}^0$ to which it belongs and its position k_i from the left at that level. As an example, the root node 0 of $\mathcal{T}^0$ is represented by $(\ell_0, 1)$. We will interchangeably use both i and (ℓ_i, k_i) to denote a node.

n	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
17	2.34	3.22	3.93	4.49	4.89	5.13	5.21
18	2.36	3.29	4.05	4.67	5.14	5.45	5.60
19	2.37	3.32	4.09	4.73	5.21	5.55	5.74
20	2.39	3.38	4.19	4.86	5.39	5.77	6.02
21	2.40	3.38	4.20	4.88	5.43	5.85	6.15
22	2.42	3.43	4.27	4.98	5.58	6.06	6.42
23	2.43	3.44	4.28	4.99	5.60	6.09	6.48
24	2.45	3.48	4.33	5.07	5.71	6.24	6.67

Table 1: The expected header lengths for $17 \leq n \leq 24$ and $2 \leq r \leq 8$ in the CSD method.

n	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
17	3.06	3.87	4.49	4.96	5.29	5.46	5.49
18	3.04	3.88	4.53	5.04	5.41	5.65	5.74
19	3.12	4.01	4.72	5.27	5.69	5.97	6.11
20	2.86	3.70	4.40	4.98	5.44	5.80	6.03
21	3.69	4.44	5.07	5.60	6.02	6.35	6.56
22	3.19	4.09	4.86	5.50	6.01	6.40	6.69
23	3.27	4.20	5.01	5.68	6.23	6.66	6.98
24	2.70	3.54	4.35	5.08	5.71	6.24	6.67

Table 2: The expected header lengths for $17 \leq n \leq 24$ (in each case $32-n$ revoked dummy users are added) and $2 \leq r \leq 8$ in the SD method.

n	$r=2$	$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$
17	2.76	3.88	4.66	5.24	5.64	5.87	5.96
18	2.67	3.76	4.53	5.09	5.51	5.78	5.92
19	2.61	3.72	4.52	5.16	5.67	6.07	6.35
20	2.56	3.66	4.48	5.15	5.69	6.12	6.44
21	2.52	3.64	4.52	5.26	5.90	6.43	6.84
22	2.49	3.62	4.53	5.31	5.99	6.56	7.03
23	2.47	3.62	4.58	5.41	6.14	6.77	7.28
24	2.45	3.60	4.59	5.45	6.19	6.83	7.34

Table 3: The expected header lengths for $17 \leq n \leq 24$ (in each case $32-n$ privileged dummy users are added) and $2 \leq r \leq 8$ in the SD method.

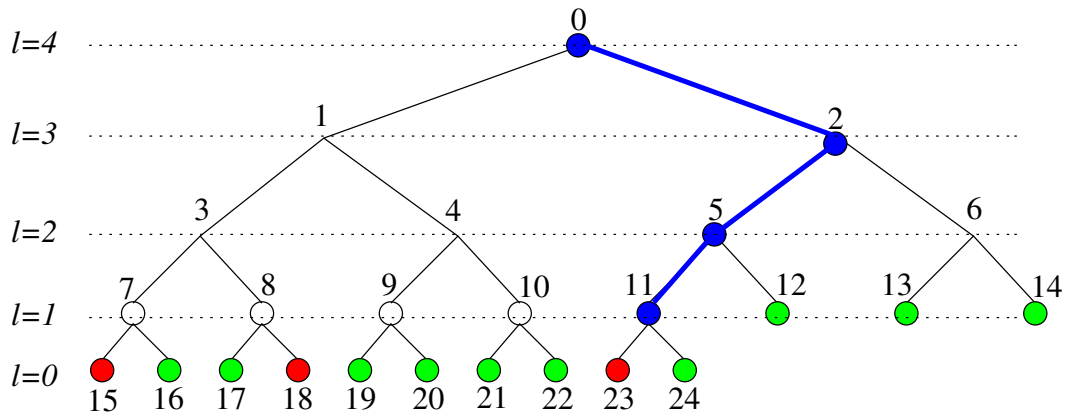

Figure 3: Level numbers and the path $\mathcal{P}^0$ (with blue) in $\mathcal{T}^0$. Nodes coloured blue are at position $k_\ell^{\mathcal{P}}$ for the respective level ℓ.

Non-full subtrees at each level of $\mathcal{T}^0$: Let us take a closer look at the structure of the tree $\mathcal{T}^0$. In case $\mathcal{T}^0$ is full, all its subtrees are also full. In case $\mathcal{T}^0$ is non-full, we observe that every level ℓ (> 0) of $\mathcal{T}^0$ can have at most one non-full subtree. To identify these subtrees, we look at the path joining the root node 0 of $\mathcal{T}^0$ with node $n-2$ (the last non-leaf node) and denote it by $\mathcal{P}_0$. There is exactly one node on $\mathcal{P}_0$ for every level ℓ (> 0) of $\mathcal{T}^0$. For level ℓ, the position of the node lying on the path $\mathcal{P}_0$ from the left, is denoted by $k_\ell^{\mathcal{P}}$. Let j be a node on $\mathcal{P}_0$, say the node represented by $(\ell, k_\ell^{\mathcal{P}})$. The part of the path $\mathcal{P}_0$ lying in the subtree $\mathcal{T}^j$ is denoted as $\mathcal{P}_j$. For the level ℓ, the subtree $\mathcal{T}^j$ rooted at node $(\ell, k_\ell^{\mathcal{P}})$ is the only possibly non-full subtree rooted at level ℓ. The subtrees to the left and right of node $k_\ell^{\mathcal{P}}$ at level ℓ are all full. The subtrees to the left (right) of node $k_\ell^{\mathcal{P}}$ of level ℓ have 2^ℓ (respectively $2^{\ell-1}$) leaves. The number of leaves in the only possibly non-full subtree rooted at level ℓ (subtree rooted at node $(\ell, k_\ell^{\mathcal{P}})$ of level ℓ) is denoted by $\lambda^{\ell,\mathcal{P}}$. Hence, $2^{\ell-1} < \lambda^{\ell,\mathcal{P}} \leq 2^\ell$. More specifically, $\lambda^{\ell,\mathcal{P}} = n - ((k_\ell^{\mathcal{P}} - 1) \times 2^\ell) - ((2^{\ell_0-\ell} - k_\ell^{\mathcal{P}}) \times 2^{\ell-1})$. Also, $k_\ell^{\mathcal{P}} = \lceil \frac{q_0}{2^\ell} \rceil$. We define $k_\ell^{\mathcal{P}_j}$ for the path $\mathcal{P}_j$ as the position of the node at level ℓ on $\mathcal{P}_j$ from the left in the subtree $\mathcal{T}^j$. Hence, $k_\ell^{\mathcal{P}}$ is also denoted as $k_\ell^{\mathcal{P}_0}$. One can see that $k_\ell^{\mathcal{P}_j} = \left\lceil \frac{q_0 - (k_{\ell_j}^{\mathcal{P}} - 1) \times (2^{\ell_j})}{2^\ell} \right\rceil = \lceil \frac{q_0}{2^\ell} \rceil - (k_{\ell_j}^{\mathcal{P}} - 1) \times (2^{\ell_j - \ell})$.

4.2 Recurrences $N(n, r, h)$ and $T(n, r, h)$

Theorem 1. *For a subtree $\mathcal{T}^i$ of $\mathcal{T}^0$ with λ_i ($2^\ell < \lambda_i \leq 2^{\ell+1}$) leaves,*

$$N(\lambda_i, r_1, h_1) = T(\lambda_i, r_1, h_1) + \sum_{j \in \mathsf{IN}(i)} T(\lambda_j, r_1, h_1 - 1) \tag{1}$$

where $\mathsf{IN}(i)$ is the set of all internal nodes in the subtree $\mathcal{T}^i$ excluding the node i.

Proof. We show that a revocation pattern is counted in $N(\lambda_i, r_1, h_1)$ if and only if it is counted in exactly one of $T(\lambda_i, r, h)$ or $T(\lambda_j, r, h-1)$ for some $j \in \mathsf{IN}(i)$. First we consider a (λ_i, r)-revocation pattern that is counted in $N(\lambda_i, r, h)$. There exists a minimal subtree $\mathcal{T}^j$ ($j \in \mathsf{IN}(i)$) of $\mathcal{T}^i$ that contains all the revoked leaves. If this subtree is rooted at i itself, then that revocation pattern is counted in $T(\lambda_i, r, h)$ and is covered by h subsets of $\mathcal{S}$. For any other node j ($\neq i$), the revocation pattern is counted in $T(\lambda_j, r, h-1)$ and has to be covered by $h-1$ subsets of $\mathcal{S}$. The rest of the $\lambda_i - \lambda_j$ privileged users form one SD subset of the cover. The total cover size will hence be h. Since a set $\mathcal{R}$ of revoked users has a corresponding unique minimal subtree $\mathcal{T}^j$ of $\mathcal{T}^i$ containing all the users in $\mathcal{R}$, hence it is counted exactly once on the right side of Equation (1).

Now, let us consider a (λ_i, r)-revocation pattern that has been counted in $T(\lambda_i, r, h)$. By the definitions of T and N, the (λ_i, r)-revocation patterns that are counted in $T(\lambda_i, r, h)$ are also counted in $N(\lambda_i, r, h)$. For some other revocation pattern, counted in $T(\lambda_j, r, h-1)$ (for some $j \in \mathsf{IN}(i)$), both subtrees of $\mathcal{T}^j$ contain at least one revoked user in each. Hence, the minimal subtree of $\mathcal{T}^i$ containing the r revoked users for such a revocation pattern is $\mathcal{T}^j$. For the revocation patterns counted in $T(\lambda_j, r, h-1)$, the privileged users of the subtree $\mathcal{T}^j$ have been covered with $h-1$ SD subsets of $\mathcal{S}$. The rest of the $\lambda_i - \lambda_j$ users are all privileged and are covered by one more SD subset $S_{i,j}$. Hence, the corresponding (λ_i, r)-revocation pattern is counted in $N(\lambda_i, r, h)$. □

Theorem 2. *For a subtree $\mathcal{T}^i$ of $\mathcal{T}^0$ with λ_i ($2^\ell < \lambda_i \leq 2^{\ell+1}$) leaves,*

$$T(\lambda_i, r_1, h_1) = \sum_{r'=1}^{r_1-1} \sum_{h'=0}^{h_1} N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h') \tag{2}$$

where λ_{2i+1} (respectively λ_{2i+2}) is the number of leaves in the left (respectively right) subtree of $\mathcal{T}^i$.

Proof. We show that a revocation pattern is counted in $T(\lambda_i, r_1, h_1)$ if and only if it is counted in the right hand side of (2). For a given λ_i, the number of leaves in the left and right subtrees get fixed to λ_{2i+1} and λ_{2i+2} respectively. When a (λ_i, r_1)-revocation pattern is counted in $T(\lambda_i, r_1, h_1)$, both the subtrees of $\mathcal{T}^i$ must have at least one revoked user. Assuming the left subtree of $\mathcal{T}^i$ has r' revoked users, the right subtree should have $r_1 - r'$ revoked users since the total number of revoked users is r_1. Similarly, assuming that the privileged users in this left subtree are covered by h' sets of $\mathcal{S}$, the privileged users in the right subtree should be covered by $h_1 - h'$ sets of $\mathcal{S}$. The number of (λ_{2i+1}, r')-revocation patterns in the left subtree covered by h' subsets is $N(\lambda_{2i+1}, r', h')$. Similarly, the number of $(\lambda_{2i+2}, r_1 - r')$-revocation patterns in the right subtree covered by $h_1 - h'$ subsets is $N(\lambda_{2i+2}, r_1 - r', h_1 - h')$. Each such (λ_{2i+1}, r')-revocation pattern in the left subtree along with a $(\lambda_{2i+2}, r_1 - r')$-revocation pattern in the right subtree gives rise to a (λ_i, r)-revocation pattern in the tree $\mathcal{T}^i$ that is covered by h_1 subsets of $\mathcal{S}$. Hence, for all values of $r' \in \{1, \ldots, r_1 - 1\}$ and all values of $h' \in \{0, \ldots, h_1\}$, $N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h')$ counts all the possible $T(\lambda_i, r_1, h_1)$.

Any (λ_i, r_1)-revocation pattern covered by h' subsets will be counted in some $N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h')$. The ones counted in $N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h')$ for fixed values of r' and h' are counted exactly once in it. For other values of r' and h', the corresponding (λ_i, r_1)-revocation patterns will be counted in the respective $N(\lambda_{2i+1}, r', h') \times N(\lambda_{2i+2}, r_1 - r', h_1 - h')$. Hence, a (λ_i, r_1)-revocation pattern is counted on the right hand side of (2) if and only if it is counted in $T(\lambda_i, r_1, h_1)$. □

Boundary conditions: The boundary conditions on $T(\lambda_i, r_1, h_1)$ and $N(\lambda_i, r_1, h_1)$ are given in Table 4. Other than the tabulated values, $N(\lambda_i, r_1, h_1) = 0$ for $\lambda_i \leq 0$ and $T(\lambda_i, r_1, h_1) = 0$ for $\lambda_i \leq 1$. From recurrences in Theorems (1) and (2) and the boundary conditions on these recurrences, one can find the value of $N(n, r, h)$ for any given n, r and h using dynamic programming.

4.3 Algorithms to compute $N(n, r, h)$ and $T(n, r, h)$

Substituting for $j \in \mathsf{IN}(i)$: To use these recurrences as an algorithm, the nodes $j \in \mathsf{IN}(i)$ in (1) for a node i have to be explicitly identified and the corresponding λ_js have to be substituted. As described in section 4.1 before, there are at most three types of subtrees rooted at a level ℓ_j of $\mathcal{T}^0$: full subtrees of height ℓ_i, full subtrees of height $\ell_i - 1$ and a non-full complete subtree of height ℓ_i.
(1) For a subtree $\mathcal{T}^i$ that is full and is of height 2^{ℓ_i} (to the left of the node at position $k_{\ell_i}^{\mathcal{P}}$ at level ℓ_i):

$$N(\lambda_i, r_1, h_1) = T(\lambda_i, r_1, h_1) + \sum_{\ell_j=1}^{\ell_i-1} (2^{\ell_i - \ell_j}) \times T(2^{\ell_j}, r_1, h_1 - 1). \tag{3}$$

$T(\lambda_i, r_1, h_1)$	$r_1 < 0$	$r_1 = 0$	$r_1 = 1$	$2 \leq r_1 < n$	$r_1 = n$	$r_1 > n$
$h_1 = 0$	0	0	0	0	1	0
$h_1 \geq 1$	0	0	0	from (2)	0	0
$N(\lambda_i, r_1, h_1)$	$r_1 < 0$	$r_1 = 0$	$r_1 = 1$	$2 \leq r_1 < n$	$r_1 = n$	$r_1 > n$
$h_1 = 0$	0	0	0	0	1	0
$h_1 = 1$	0	1	n	from (1)	0	0
$h_1 > 1$	0	0	0	from (1)	0	0

Table 4: Boundary conditions on $T(n, r, h)$ and $N(n, r, h)$.

(2) For a subtree $\mathcal{T}^i$ that is full and is of height $2^{\ell_i - 1}$ (to the right of the node at position $k^{\mathcal{P}}_{\ell_i}$ at level ℓ_i):

$$N(\lambda_i, r_1, h_1) = T(\lambda_i, r_1, h_1) + \sum_{\ell_j=2}^{\ell_i - 1} (2^{\ell_i - \ell_j}) \times T(2^{\ell_j - 1}, r_1, h_1 - 1). \tag{4}$$

(3) For the only possibly non-full subtree $\mathcal{T}^i$ for $i = (\ell_i, k^{\mathcal{P}}_{\ell_i})$ of height 2^{ℓ_i} (at position $k^{\mathcal{P}}_{\ell_i}$ at level ℓ_i):

$$\begin{aligned} N(\lambda_i, r_1, h_1) = T(\lambda_i, r_1, h_1) & \\ + \sum_{\ell_j=2}^{\ell_i - 1} [(k^{\mathcal{P}_i}_{\ell_j} - 1) \times T(2^{\ell_j}, r_1, h_1 - 1) + T(\lambda^{\ell_j, \mathcal{P}}, r_1, h_1 - 1) & \\ + (2^{\ell_i - \ell_j} - k^{\mathcal{P}_i}_{\ell_j}) \times T(2^{\ell_j - 1}, r_1, h_1 - 1)]. & \end{aligned} \tag{5}$$

Dynamic Programming: Computing $N(n, r, h)$ and $T(n, r, h)$ requires computing $N(\lambda_i, r_1, h_1)$ and $T(\lambda_i, r_1, h_1)$ for some smaller λ_i, r_1 and h_1. We use dynamic programming technique where all values of $N(\lambda_i, r_1, h_1)$ and $T(\lambda_i, r_1, h_1)$ for smaller λ_i, r_1 and h_1 are pre-computed. The algorithm to compute $T(n, r, h)$ from these pre-computed values is obtained from (2) in a straight forward manner. The algorithm to compute $N(n, r, h)$ from these pre-computed values is obtained from (1) (more specifically from either of (3) or (5)). Level ℓ_i of $\mathcal{T}^0$ has $k^{\mathcal{P}}_{\ell_i} - 1$ full subtrees of height ℓ_i, $(2^{\ell_0 - \ell_i}) - k^{\mathcal{P}}_{\ell_i}$ full subtrees of height $\ell_i - 1$ and one possibly non-full subtree. For every level in the tree $\mathcal{T}^0$, $T(\lambda_i, r, h - 1)$ is pre-computed once for each of the three types of nodes and used to compute $N(n, r, h)$.

Space and Time complexity of the algorithm: Using (2) to compute $T(n, r, h)$ from the pre-computed values of $N(\cdot, \cdot, \cdot)$ requires $O(rh)$ memory operations and multiplications. Equation (1) shows how $N(n, r, h)$ is related to pre-computed values of $T(\cdot, \cdot, \cdot)$. Actual computation is done using (3), (4) and (5). This requires $O(1)$ memory operations and a single addition for each of the $\lceil \log n \rceil$ levels of $\mathcal{T}^0$. Hence, the time complexity for computing $T(n, r, h)$ and then $N(n, r, h)$ *from pre-computed values* is $O(rh + \log n)$.

These pre-computed values in turn need to be computed. By the form of (3), (4) and (5) there are $\log n$ subtrees to be considered. For each such subtree, $O(rh)$ values need to be computed and the computation of these will be based on values computed earlier. A dynamic programming algorithm proceeds in a bottom-up fashion by computing the $O(rh)$ values corresponding to smaller sub-trees and then using these to compute the values for progressively larger sub-trees. This takes a total of $O(r^2 h^2 \log n + rh \log^2 n)$ time. The space requirement is given by the number of pre-computed values that need to be stored to compute $N(n, r, h)$. For each of the $O(\log n)$ sub-trees, a total of $O(rh)$ values need to be stored and so the space complexity is $O(rh \log n)$.

n	r	h	$N(n,r,h)$
126	63	37	7.44×10^{35}
127	61	37	1.27×10^{36}
128	64	37	2.96×10^{36}
131	65	38	2.33×10^{37}

Table 5: Listing a few values of n, r, h and their corresponding $N(n,r,h)$.

The above time and space complexities are required for a single set of values of n, r and h. For a fixed n and r, it may be required to compute the values of $N(n,r,h)$ for all possible values of h. This would be a typical requirement for a broadcast centre which will have a fixed number of users and for a particular transmission knows the number of revoked users. The corresponding time and space complexities can be obtained by substituting an appropriate value for h. A trivial bound is n; but, later we show that $h \leq 2r-1$ which gives the expressions $O(r^4 \log n + r^2 \log n)$ and $O(r^2 \log^2 n)$ for time and space complexities respectively. For large n and moderate values of r, these are practical complexities.

Further, allowing r to range over all the $O(n)$ possible values leads to $O(n^4 \log n + n^2 \log^2 n)$ time and $O(n^2 \log n)$ space complexities respectively. If we are interested in computing $N(i,r,h)$ for all $2 \leq i \leq n$ and all possible values of r and h, then the time and space complexities are $O(n^5 + n^3 \log n)$ and $O(n^3)$ respectively.

Table 5 lists the values of n, r, h and their corresponding $N(n,r,h)$ for some values of n, r and h. Note that computing these values would not be possible by direct enumeration. For example, attempting direct enumeration to tackle the first row of Table 5, would require considering $\binom{126}{63}$ possible revocation patterns which is way beyond the present computational capabilities.

4.4 Upper Bounds on Header Length of the (C)SD method

We first prove that the header length of the CSD scheme is upper bounded by $2r-1$. For full trees of the SD method, this bound was proved in [NNL01].

Theorem 3. *$N(\lambda_i, r_1, h_1) = 0$ when $h_1 > 2r_1 - 1$. $T(\lambda_i, r_1, h_1) = 0$ when $h_1 \geq 2r_1 - 1$.*

Proof. First we show that $T(\lambda_i, r_1, h_1) = 0$ when $h_1 \geq 2r_1 - 1$ in (1). We prove this from (2) by induction on r_1. The boundary conditions have been listed in Table 4. We know that, $2^{\ell_i - 1} < \lambda_i \leq 2^{\ell_i}$. By induction hypothesis, when $h' > 2r' - 1$ and $1 \leq r' < r_1$, $N(\lambda_{2i+1}, r', h') = 0$. If $h' \leq 2r' - 1$, then $h_1 - h' > 2r_1 - 1 - h' \geq 2r_1 - 1 - 2r' + 1 = 2(r_1 - r')$. Then, again by induction hypothesis, $N(\lambda_{2i+2}, r_1 - r', h_1 - h') = 0$. Hence, when $h_1 \geq 2r_1 - 1$, $T(\lambda_i, r_1, h_1) = 0$.

Now, if $h_1 > 2r_1 - 1$, the other terms on the right hand side of (1) are $T(\lambda_i, r_1, h_1 - 1)$ where $h_1 - 1 \geq 2r_1 - 1$ for all terms and hence are all 0 as proved above. Hence, when $h_1 > 2r_1 - 1$, $N(\lambda_i, r_1, h_1) = 0$. □

We later show that for sufficiently large n, $N(n, r, 2r-1)$ is positive and also characterize the minimum n for which this happens.

Next, we show that $N(n,r,h)$ is monotonic on n for fixed r and h. While intuitive, there does not seem to be an easy way to prove this.

Lemma 4. *Let $n_1 \geq n_2$. If $N(n_2, r, h) \neq 0$ then $N(n_1, r, h) \neq 0$. If $T(n_2, r, h) \neq 0$ then $T(n_1, r, h) \neq 0$.*

Proof. Let $T(n_2, r, h) \neq 0$. From (2) we get:

$$T(n_2, r, h) = \sum_{r'=1}^{r-1} \sum_{h'=0}^{h} N(\lambda_1, r', h') \times N(\lambda_2, r - r', h - h').$$

Let $RH = \{(r_1, h_1) \ldots, (r_s, h_s)\}$ be such that both $N(\lambda_1, r', h')$ and $N(\lambda_2, r - r', h - h')$ are non-zero (and hence $N(\lambda_1, r', h') \times N(\lambda_2, r - r', h - h')$ is non-zero) when $(r', h') \in RH$. Hence, we can also write:

$$T(n_2, r, h) = \sum_{(r',h') \in RH} N(\lambda_1, r', h') \times N(\lambda_2, r - r', h - h').$$

Since $\lambda_1 < n_2$ (by the structure of $\mathcal{T}^0$ with n_2 leaves), hence by induction hypothesis, for any $\lambda \geq \lambda_1$, $N(\lambda_1, r, h) \neq 0$ implies $N(\lambda, r, h) \neq 0$. Similarly, since $\lambda_2 < n_2$, hence by induction hypothesis, for any $\lambda \geq \lambda_2$, $N(\lambda_2, r, h) \neq 0$ implies $N(\lambda, r, h) \neq 0$. When there are n_1 leaves in the tree let there be λ'_1 leaves in the left subtree and λ'_2 leaves in the right subtree of the root node. Hence, by the construction of $\mathcal{T}^0$, we get $\lambda'_1 \geq \lambda_1$ and $\lambda'_2 \geq \lambda_2$. In the expression for $T(n_1, r, h)$, for $(r', h') \in RH$, by induction hypothesis, $N(\lambda'_1, r', h')$ and $N(\lambda'_2, r - r', h - h')$ are both non-zero. Hence, for at least $(r', h') \in RH$, $N(\lambda'_1, r', h') \times N(\lambda'_2, r - r', h - h')$ is non-zero. Thus, $T(n_1, r, h) \neq 0$.

Now, let $N(n_2, r, h) \neq 0$. From (1) we get:

$$N(n_2, r, h) = T(n_2, r, h) + \sum_{j=1}^{n_2-2} T(\lambda_j, r, h - 1).$$

Let $I = \{i_1, \ldots, i_t\}$ be the nodes of $\mathcal{T}^0$ (with n_2 leaves) such that $T(\lambda_i, r, h) \neq 0$ for $i \in I$. By induction hypothesis, for any $\lambda_j < n_2$ and $\lambda_i > \lambda_j$, if $T(\lambda_j, r, h) \neq 0$ then $T(\lambda_i, r, h) \neq 0$. Hence, we can also write:

$$N(n_2, r, h) = T(n_2, r, h) + \sum_{i \in I} T(\lambda_i, r, h - 1).$$

Here, $T(n_2, r, h) \neq 0$ implies $T(n_1, r, h) \neq 0$ by the first part of this proof. By the construction of the tree $\mathcal{T}^0$, $\lambda'_i \geq \lambda_i$ where λ'_i is the number of leaves (users) in the subtree rooted at node i of the tree $\mathcal{T}^0$ for n_1 users. By induction hypothesis, at least for $i \in I$, since $T(\lambda_i, r, h-1) \neq 0$, hence $T(\lambda'_i, r, h-1) \neq 0$. Thus, $N(n_1, r, h) \neq 0$. □

Now, we prove that if r is not small compared to n, then $T(n, r, 2r - 2) = 0$.

Lemma 5. *For $n \leq 2^{2k+1}$ and $r > 2^k$, $T(n, r, 2r - 2) = 0$.*

Proof. For $T(n, r, 2r - 2)$ in (2), let $h' < 2r' - 1$, then $h - h' = 2r - 2 - h' > 2r - 2 - 2r' + 1 = 2(r - r') - 1$. Hence by Theorem 3, $N(\lambda_2, r - r', h - h') = 0$. Similarly, if $h' > 2r' - 1$, $N(\lambda_1, r', h') = 0$. So, in the expression for $T(n, r, 2r - 2)$, the terms on the right hand side of (2) are 0 if $h' \neq 2r' - 1$. Hence,

$$T(n, r, 2r - 2) = \sum_{r'=1}^{r-1} N(\lambda_1, r', 2r' - 1) \times N(\lambda_2, r - r', 2(r - r') - 1). \tag{6}$$

Now by induction on λ_i, we prove that $N(\lambda_1, r', 2r' - 1) = 0$ and $N(\lambda_2, r - r', 2(r - r') - 1) = 0$. The boundary conditions have been listed in Table 4. By induction hypothesis, for $\lambda_i \leq 2^{2m+1}$ where $m < k$ and $r' > 2^m$ let us assume $T(\lambda_i, r', 2r'-2) = 0$. In (6), let $r' \geq \frac{r}{2}$ which implies $r' > 2^{k-1}$. Hence, for $\lambda_i \leq 2^{2k-1}$, $T(\lambda_i, r', 2r'-2) = 0$ by the induction hypothesis. Also, by Theorem 3, $T(\lambda_1, r', 2r' - 1) = 0$. Putting these values in (1), we get $N(\lambda_1, r', 2r' - 1) = 0$. Similarly, for $r - r' \geq \frac{r}{2}$ which implies $r - r' > 2^{k-1}$, we get $N(\lambda_2, r - r', 2(r - r') - 1) = 0$. Hence, from (6) $T(n, r, 2r - 2) = 0$. □

Some insight: Given a revocation pattern, if we revoke one more user from it, that can result in either increase, decrease or no change in the cover size. Increase in cover size mostly happens when the newly revoked user is not adjacent to any previously revoked user. The cover size remains unchanged or decreases when the newly revoked user is adjacent to a previously revoked user. Decrease in cover size happens when the user in a singleton subset of the cover is revoked. As the number of revoked users increase, the maximum possible cover size for that number of revoked users increases up to a certain point. After that the maximum possible cover size decreases. One may also observe that for $n > 2$ ($\ell_1 \geq 1$), $q_0/2 = n - 2^{\ell_1}$. Since 2^{ℓ_1} is even for $\ell_1 \geq 1$, hence when n is even $q_0/2$ is even and when n is odd $q_0/2$ is odd.

Theorem 6. *The header length in the CSD method for n users is at most $\lfloor \frac{n}{2} \rfloor$ irrespective of the number of revoked users.*

Proof. First, we show that $N(n, r, h) = 0$ for $h > \frac{n}{2}$ for any r. We prove this by induction on n. From (1) we have:

$$N(n, r, h) = T(n, r, h) + \sum_{i=1}^{n-2} T(\lambda_i, r, h-1)$$

and hence, $T(n, r, h) \leq N(n, r, h)$. When $\lambda_i < n$ and $h-1 \geq \lfloor \frac{n}{2} \rfloor$, $N(\lambda_i, r, h-1) = 0$. Thus, $\sum_{i=1}^{n-2} T(\lambda_i, r, h-1) = 0$. From (2) we get:

$$T(n, r, h) = \sum_{r'=1}^{r-1} \sum_{h'=0}^{h} N(\lambda_1, r', h') \times N(\lambda_2, r - r', h - h').$$

When $h' > \frac{\lambda_1}{2}$, $N(\lambda_1, r', h') = 0$ by induction hypothesis. When $h' \leq \frac{\lambda_1}{2}$, since $h > \frac{n}{2}$, $h - h' > \frac{n}{2} - \frac{\lambda_1}{2} = \frac{\lambda_2}{2}$. Therefore, $N(\lambda_2, r - r', h - h') = 0$ by induction hypothesis. Hence, $N(n, r, h) = 0$ for $h > \frac{n}{2}$ for any r.

Next, we show that the upper bound of $\lfloor \frac{n}{2} \rfloor$ is actually achieved. First let us assume that n is even and hence $q_0/2$ is even. We construct a revocation pattern such that none of the users are revoked initially. Now, let us form a revocation pattern by revoking one user from each of the $q_0/2$ subtrees rooted at level q_1 with leaves at level q_0 and one user each from subtrees rooted at level 2 with leaves at level 1. Since all the privileged users would form singleton subsets in the cover for this revocation pattern, hence the header length for the revocation pattern thus constructed is of size q_1 ($= \frac{n}{2}$). Now, if we attempt to revoke any other user, then by pigeonhole principle, one of the sets in the cover gets removed and hence the header length decreases. Hence, for even n, the maximum header length is $\frac{n}{2}$.

For odd n, $q_0/2$ is odd. We construct a revocation pattern similarly by revoking one user from each of the $q_0/2$ subtrees rooted at level q_1 with leaves at level q_0 and one user each from subtrees rooted at level 2 with leaves at level 1. Since $q_0/2$ is odd, there will be one subtree (rooted at the node at position $k_2^{\mathcal{P}}$) with leaves at both levels 0 and 1. For this subtree, only one out of the three users in it is revoked. Since all the privileged users would form singleton subsets (except the one generated from the above subtree) in the cover for this revocation pattern, hence the cover size for the revocation pattern thus constructed is of size q_1 ($= \lfloor \frac{n}{2} \rfloor$). This is again the maximum header length by the same argument as above.

Hence, the maximum header length is $\lfloor \frac{n}{2} \rfloor$ for n users. □

Theorem 6 gives a bound on the header size. Previously, the bound of $2r - 1$ for the header size had been obtained. The next result completes the picture by obtaining yet another bound on the header size.

Theorem 7. *The maximum header length in the CSD method for n users is* $\min(2r - 1, \lfloor \frac{n}{2} \rfloor, n - r)$.

Proof. The bounds $2r - 1$ and $\lceil n/2 \rceil$ have already been shown. We show the bound of $n - r$ on the header size. The proof of this is similar to the first part of the proof of Theorem 6, i.e., we show that $N(n, r, h) = 0$ for $h > n - r$.

For $\lambda_i < n$, we have $h - 1 > n - 1 - r \geq \lambda_i - r$ and hence using induction, $N(\lambda_i, r, h-1) = 0$ which implies that $T(\lambda_i, r, h-1)$ is also zero. Again, consider the value of $T(n, r, h)$ and the recurrence expressing this in terms of $N(\lambda_1, r', h')$ and $N(\lambda_2, r-r', h-h')$, where $\lambda_1 + \lambda_2 = n$. If $h' > \lambda_1 - r'$, then using induction, $N(\lambda_1, r', h') = 0$. So, suppose that $h' \leq \lambda_1 - r'$. Using $h > n - r$, we have $h - h' > (n - \lambda_1) - (r - r') = \lambda_2 - (r - r')$ and again using induction, $N(\lambda_2, r - r', h - h') = 0$.

This shows that $T(n, r, h) = 0$ which combined with the fact that the other relevant values of $T(\cdot, \cdot, \cdot)$ are zero, shows that $N(n, r, h) = 0$ for $h > n - r$. □

The bound given by Theorem (7) gives a complete picture. If $r \leq n/4$, then the bound $2r - 1$ is appropriate; if $n/4 < r \leq n/2$, then the bound $\lceil n/2 \rceil$ is appropriate; and for $r > n/2$, the bound $(n - r)$ is appropriate. The last bound has an important consequence. If the number of revoked users is greater than $n/2$, it may appear that using individual transmission to the privileged users would be better than using the CSD method. But, The bound of $(n - r)$ on the header size shows that this is not true. Using the CSD method is never worse than individual transmission to privileged users.

The bound of Theorem 7 hold for the SD scheme, i.e., for full trees. We note that the only previously *proved* upper bound for the SD scheme is $2r - 1$. The other two bounds bounds, while intuitive, do not appear to have been reported with proofs in the literature. In fact, there does not seem to be an easy way to argue about these bounds without using the recurrences that we have derived.

Definition 1. *n_r: For a given r, n_r is the minimum number of users required to be in the system so that there exists an (n, r)-revocation pattern covered by a header length of $2r - 1$. In other words, n_r is the minimum n for which $N(n, r, 2r - 1) > 0$.*

Theorem 3 shows that the upper bound on the header length is $2r - 1$. By characterizing n_r, we show that this upper bound on h is actually achieved.

Lemma 8. *In the CSD method, $2^{k-1} < r \leq 2^k$ if and only if $2^{2k} < n_r < 2^{2k+1}$.*

Proof. We first prove that if $2^{k-1} < r \leq 2^k$, then $2^{2k} < n_r \leq 2^{2k+1}$ (by showing that $N(2^{2k}, r, 2r - 1) = 0$ and $N(2^{2k+1}, r, 2r - 1) \neq 0$). Although by Theorem 3, $T(2^{2k+1}, r, 2r - 1) = 0$, we show that $T(2^{2k}, r, 2r - 2) \neq 0$ and hence at least one of the terms on the right hand side of (1) is non-zero and hence $N(2^{2k+1}, r, 2r - 1) \neq 0$. From (2) we get:

$$T(2^{2k}, r, 2r - 2) = \sum_{r'=1}^{r-1} \sum_{h'=0}^{2r-2} N(2^{2k-1}, r', h') \times N(2^{2k-1}, r - r', 2r - 2 - h').$$

When $h' > 2r' - 1$, $N(2^{2k-1}, r', h') = 0$ by Theorem 3. Similarly, when $h' < 2r' - 1$, $2r - 2 - h' > 2r - 2 - 2r' + 1 = 2(r - r') - 1$ and hence $N(2^{2k-1}, r - r', 2r - 2 - h') = 0$. Hence, we get

$$T(2^{2k}, r, 2r - 2) = \sum_{r'=1}^{r-1} N(2^{2k-1}, r', 2r' - 1) \times N(2^{2k-1}, r - r', 2(r - r') - 1).$$

When $r' = \lceil \frac{r}{2} \rceil$ ($2^{k-2} < r' \leq 2^{k-1}$) by induction hypothesis, $n_{r'} \leq 2^{2k-1}$ and hence by Lemma 4, both $N(2^{2k-1}, r', 2r' - 1)$ and $N(2^{2k-1}, r - r', 2(r - r') - 1)$ are non-zero. Hence, $T(2^{2k}, r, 2r - 2) \neq 0$ which implies $N(2^{2k+1}, r, 2r - 1) \neq 0$. Since $T(n_r, r, 2r - 1) = 0$ and $T(2^{2k-1}, r, 2r - 2) = 0$ hence, $n_r < 2^{2k+1}$. Next, we show that $N(2^{2k}, r, 2r - 1) = 0$. By Theorem 3, $T(2^{2k}, r, 2r - 1) = 0$. By Lemma 5, for all $\lambda_i \leq 2^{2k-1}$ and $r > 2^{k-1}$, $T(\lambda_i, r, 2r - 2) = 0$ and hence $N(2^{2k}, r, 2r - 1) = 0$.

Next, we prove that for some $2^{2k} < n_r \leq 2^{2k+1}$, the corresponding r is such that $2^{k-1} < r \leq 2^k$. Let the corresponding r be such that $2^{k'-1} < r \leq 2^{k'}$ where $k \neq k'$. Then by the argument above, we know that $2^{2k'} < n_r \leq 2^{2k'+1}$ which is a contradiction since n_r is unique for a given r by definition. Hence the corresponding r is such that $2^{k-1} < r \leq 2^k$. □

r	1	2	3	4	5	6	7	8
n_r	2	6	18	22	66	70	82	86

Table 6: Listing a few values of r and their corresponding n_r.

One may note here that not all n between 2^{k-1} and 2^k is an n_r for some r. We list a few r and their corresponding n_r in Table 6. Next, we find an expression for n_r.

Theorem 9. *In the CSD method, let $2^{k-1} < r \leq 2^k$. When $r \leq 2^{k-1} + 2^{k-2}$, let $r_1 = 2^{k-2}$ and $r_0 = r - 2^{k-2}$ and hence,*

$$n_r = n_{r_0} + 2^{2k-2} + 2^{2k-1}$$

and when $r > 2^{k-1} + 2^{k-2}$, let $r_0 = 2^{k-1}$ and $r_1 = r - 2^{k-1}$ and hence,

$$n_r = 2^{2k-1} + n_{r_1} + 2^{2k-1}.$$

Proof. From Lemma 8 we know that for $2^{k-1} < r \leq 2^k$, $2^{2k} < n_r \leq 2^{2k+1}$. For such an n_r, $\lambda_1 = n_r - 2^{2k-1}$ and $\lambda_2 = 2^{2k-1}$. From (1) we get

$$N(n_r, r, 2r-1) = T(n_r, r, 2r-1) + T(n_r - 2^{2k-1}, r, 2r-2) + T(2^{2k-1}, r, 2r-2) + \sum_{i=3}^{n_r-2} T(\lambda_i, r, 2r-2).$$

From Theorem 3 we know that $T(n_r, r, 2r-1) = 0$. From Lemma 5 we know that when $r > 2^{k-1}$ and $\lambda_i \leq 2^{2k-1}$, $T(\lambda_i, r, 2r-2) = 0$. Hence the only non-zero component is $T(n_r - 2^{2k-1}, r, 2r-2)$. From (2) we get

$$N(n_r, r, 2r-1) = T(n_r - 2^{2k-1}, r, 2r-2) = \sum_{r'=1}^{r-1} \sum_{h'=0}^{2r-2} N(\lambda_3, r', h') \times N(\lambda_4, r - r', 2r - 2 - h').$$

By an argument similar to the one used in the proof for Lemma 8, we get

$$N(n_r, r, 2r-1) = T(n_r - 2^{2k-1}, r, 2r-2) = \sum_{r'=1}^{r-1} N(\lambda_3, r', 2r'-1) \times N(\lambda_4, r - r', 2(r-r') - 1).$$

By the construction of the complete tree $\mathcal{T}^0$ and the fact that $\mathcal{T}^2$ does not have any revoked user ($T(2^{2k-1}, r, 2r-2) = 0$) it can be seen that $2^{2k-2} < \lambda_3 \leq 2^{2k-1}$ and $2^{2k-2} \leq \lambda_4 < 2^{2k-1}$.

When $r \leq 2^{k-1} + 2^{k-2}$, let $r' = r_0 = r - 2^{k-2}$ and $r - r' = r_1 = 2^{k-2}$. From the construction of the complete tree $\mathcal{T}^0$ for $(n_{r_0} + 2^{2k-2} + 2^{2k-1})$ users, it can be seen that $\lambda_3 = n_{r_0}$ and $\lambda_4 = 2^{2k-2}$. Hence, $N(\lambda_3, r', 2r'-1) = N(n_{r_0}, r_0, 2r_0 - 1) \neq 0$ by the definition of n_r. Also, from Lemma 4 and Lemma 8 we know that for $r = 2^k$ (consequently $n_r < 2^{2k+1}$) and $\lambda \geq 2^{2k+1}$, $N(\lambda, r, 2r-1) \neq 0$. So for $r_1 = r - r' = 2^{k-2}$ and $\lambda_4 = 2^{2(k-2)+2}$ we get, $N(\lambda_4, r - r', 2(r-r') - 1) = N(2^{2k-2}, r_1, 2r_1 - 1) \neq 0$. Hence, for $r \leq 2^{k-1} + 2^{k-2}$, $N(n_r, r, 2r-1) \neq 0$ where $n_r = n_{r_0} + 2^{2k-2} + 2^{2k-1}$.

Now, we show that for $2^{k-1} < r \leq 2^{k-1} + 2^{k-2}$ ($r_0 = r - 2^{k-2}$ and $r_1 = 2^{k-2}$), $N(n_r - 1, r, 2r-1) = 0$. In the tree $\mathcal{T}^0$ for $(n_{r_0} + 2^{2k-2} + 2^{2k-1}) - 1$ users, $\lambda_3 = n_{r_0} - 1$ and $\lambda_4 = 2^{2k-2}$. Since there are $n_{r_0} - 1$ users in $\mathcal{T}^3$, at most $r_0 - 1$ revoked users can be accommodated in $\mathcal{T}^3$ so that $N(\lambda_3, r', 2r'-1) \neq 0$ and hence $r' = r_0 - 1$ and $r - r' = 2^{k-2} + 1$. By Lemma 8 for $r - r' > 2^{k-2}$, $n_{r-r'} > 2^{2k-2}$. But, $\lambda_4 = 2^{2k-2}$ and hence $N(\lambda_4, r - r', 2(r-r') - 1) = 0$. Consequently, we get $N(n_r - 1, r, 2r-1) = 0$.

When $r > 2^{k-1} + 2^{k-2}$, let $r' = r_0 = 2^{k-1}$ and $r - r' = r_1 = r - 2^{k-1}$. From the construction of the complete tree $\mathcal{T}^0$ for $(2^{2k-1} + n_{r_1} + 2^{2k-1})$ users, it can be seen that $\lambda_3 = 2^{2k-1}$ and $\lambda_4 = n_{r_1}$. Hence, $N(\lambda_4, r - r', 2(r - r') - 1) = N(n_{r_1}, r_1, 2r_1 - 1) \neq 0$ by the definition of n_r. From Lemma 4 and Lemma 8 we know that for $r = 2^k$ (consequently $n_r < 2^{2k+1}$) and $\lambda \geq 2^{2k+1}$, $N(\lambda, r, 2r - 1) \neq 0$. So for $r_0 = r' = 2^{k-1}$ and $\lambda_3 = 2^{2(k-1)+1}$ we get, $N(\lambda_3, r', 2r'-1) = N(2^{2k-1}, r_0, 2r_0-1) \neq 0$. Hence, for $r > 2^{k-1}+2^{k-2}$, $N(n_r, r, 2r-1) \neq 0$ where $n_r = 2^{2k-1} + n_{r_1} + 2^{2k-1}$.

Now, we show that for $r > 2^{k-1} + 2^{k-2}$ ($r_0 = 2^{k-1}$ and $r_1 = r - 2^{k-1}$), $N(n_r - 1, r, 2r - 1) = 0$. In the tree $\mathcal{T}^0$ for $(2^{2k-1} + n_{r_1} + 2^{2k-1}) - 1$ users, $\lambda_3 = 2^{2k-1}$ and $\lambda_4 = n_{r_1} - 1$. Since there are $n_{r_1} - 1$ users in $\mathcal{T}^4$, at most $r_1 - 1$ revoked users can be accommodated in $\mathcal{T}^4$ so that $N(\lambda_4, r - r', 2(r - r') - 1) \neq 0$ and hence $r - r' = r_1 - 1$ and $r' = 2^{k-1} + 1$. By Lemma 8 for $r' > 2^{k-1}$, $n_{r'} > 2^{2k-1}$. But, $\lambda_3 = 2^{2k-1}$ and hence $N(\lambda_3, r', 2r' - 1) = 0$. Consequently, we get $N(n_r - 1, r, 2r - 1) = 0$.

□

Note that for the SD method, when $2^{k-1} < r \leq 2^k$, the minimum n for which the header length of $2r - 1$ occurs is 2^{2k+1}. This has been earlier proved in [PB06]. It is because the CSD method allows arbitrary number of users, the value of n for which the maximum header length of $2r - 1$ is achieved for a given r is less than that of the SD method.

4.5 Generating Function from Recurrences of the CSD scheme for Full Binary trees

Let the number of users be $n = 2^{\ell_0}$ and hence the tree $\mathcal{T}^0$ is full and of height ℓ_0. For a full tree $\mathcal{T}^0$, all subtrees $\mathcal{T}^i$ are full and at level ℓ, there are $2^{\ell_0-\ell}$ subtrees with 2^ℓ leaves in each. We define $T_\ell(r, h) = T(2^\ell, r, h)$ and $N_\ell(r, h) = N(2^\ell, r, h)$. Then the recurrences (1) and (2) for counting the number of revocation patterns become:

$$N_{\ell_0}(r, h) = T_{\ell_0}(r, h) + \sum_{\ell=1}^{\ell_0-1} \left(2^{\ell_0-\ell} \times T_\ell(r, h - 1)\right). \tag{7}$$

$$T_{\ell_0}(r, h) = \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1, h_1) \times N_{\ell_0-1}(r - r_1, h - h_1) \tag{8}$$

where r is the number of revoked users.

Theorem 10. *The generating function for the sequence $N_{\ell_0}(r, h)$ of numbers defined in (7) above, is given by $X_{\ell_0}(x, y)$ where*

$$X_{\ell_0}(x, y) = \left(X_{\ell_0-1}(x, y) - xy^{2^{\ell_0-1}}\right)^2 + xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + \sum_{\ell=1}^{\ell_0-1} \left(2^{\ell_0-\ell}xy^{2^{\ell_0}-2^\ell} \times \left(X_{\ell-1}(x, y) - xy^{2^{\ell-1}}\right)^2\right).$$

Proof. Let $Y_{\ell_0}(x, y)$ be the generating function for the sequence $T_{\ell_0}(2^{\ell_0} - r, h)$ defined as follows:

$$Y_{\ell_0}(x, y) = \sum_{r=0}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} T_{\ell_0}(r, h)x^h y^{2^{\ell_0}-r} \tag{9}$$

and the generating function $X_{\ell_0}(x,y)$ for the sequence $N_{\ell_0}(2^{\ell_0}-r,h)$ can be written as:

$$X_{\ell_0}(x,y)=\sum_{r=0}^{2^{\ell_0}}\sum_{h=0}^{2^{\ell_0}-r}N_{\ell_0}(r,h)x^h y^{2^{\ell_0}-r}. \tag{10}$$

By definition, when $\ell_0=0$, $Y_0(x,y)=0$ and $X_0(x,y)=1+xy$ and when $\ell_0=1$, $Y_1(x,y)=1$ and $X_1(x,y)=1+2xy+xy^2$.

Now, we note that:

$$\begin{aligned}X^2_{\ell_0-1}(x,y)=&\Big(\sum_{r_1=0}^{2^{\ell_0-1}}\sum_{h_1=0}^{2^{\ell_0-1}-r_1}N_{\ell_0-1}(r_1,h_1)x^{h_1}y^{2^{\ell_0-1}-r_1}\Big)\times\Big(\sum_{r_2=0}^{2^{\ell_0-1}}\sum_{h_2=0}^{2^{\ell_0-1}-r_2}N_{\ell_0-1}(r_2,h_2)x^{h_2}y^{2^{\ell_0-1}-r_2}\Big)\\ =&\Big(N_{\ell_0-1}(0,1)xy^{2^{\ell_0-1}}+\sum_{r_1=1}^{2^{\ell_0-1}}\sum_{h_1=0}^{2^{\ell_0-1}-r_1}N_{\ell_0-1}(r_1,h_1)x^{h_1}y^{2^{\ell_0-1}-r_1}\Big)\times\\ &\Big(N_{\ell_0-1}(0,1)xy^{2^{\ell_0-1}}+\sum_{r_2=1}^{2^{\ell_0-1}}\sum_{h_2=0}^{2^{\ell_0-1}-r_2}N_{\ell_0-1}(r_2,h_2)x^{h_2}y^{2^{\ell_0-1}-r_2}\Big)\end{aligned} \tag{11}$$

Putting $N_{\ell_0-1}(0,1)=1$ in (11) we get:

$$\begin{aligned}X^2_{\ell_0-1}(x,y)=&\Big(\sum_{r_1=1}^{2^{\ell_0-1}}\sum_{h_1=0}^{2^{\ell_0-1}-r_1}N_{\ell_0-1}(r_1,h_1)x^{h_1}y^{2^{\ell_0-1}-r_1}\Big)\times\Big(\sum_{r_2=1}^{2^{\ell_0-1}}\sum_{h_2=0}^{2^{\ell_0-1}-r_2}N_{\ell_0-1}(r_2,h_2)x^{h_2}y^{2^{\ell_0-1}-r_2}\Big)\\ &+xy^{2^{\ell_0-1}}\Big(\sum_{r_2=1}^{2^{\ell_0-1}}\sum_{h_2=0}^{2^{\ell_0-1}-r_2}N_{\ell_0-1}(r_2,h_2)x^{h_2}y^{2^{\ell_0-1}-r_2}\Big)\\ &+xy^{2^{\ell_0-1}}\Big(\sum_{r_1=1}^{2^{\ell_0-1}}\sum_{h_1=0}^{2^{\ell_0-1}-r_1}N_{\ell_0-1}(r_1,h_1)x^{h_1}y^{2^{\ell_0-1}-r_1}\Big)\\ &+x^2y^{2^{\ell_0}}\end{aligned} \tag{12}$$

Let

$$\begin{aligned}C_{\ell_0}(x,y)=&\Big(\sum_{r_1=1}^{2^{\ell_0-1}}\sum_{h_1=0}^{2^{\ell_0-1}-r_1}N_{\ell_0-1}(r_1,h_1)x^{h_1}y^{2^{\ell_0-1}-r_1}\Big)\times\Big(\sum_{r_2=1}^{2^{\ell_0-1}}\sum_{h_2=0}^{2^{\ell_0-1}-r_2}N_{\ell_0-1}(r_2,h_2)x^{h_2}y^{2^{\ell_0-1}-r_2}\Big)\\ =&\sum_{r=2}^{2^{\ell_0}}\sum_{h=0}^{2^{\ell_0}-r}x^h y^{2^{\ell_0}-r}\sum_{r_1=1}^{r-1}\sum_{h_1=0}^{h}N_{\ell_0-1}(r_1,h_1)\times N_{\ell_0-1}(r-r_1,h-h_1).\end{aligned} \tag{13}$$

Now we take a closer look at the generating function $Y_{\ell_0}(x,y)$ of (9):

$$\begin{aligned}
Y_{\ell_0}(x,y) &= \sum_{r=0}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} T_{\ell_0}(r,h) x^h y^{2^{\ell_0}-r} \\
&= \sum_{r=0}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) \\
&= \sum_{r=0}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) \\
&= \sum_{r=2}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) \\
&+ \sum_{r=0}^{1} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) \\
&= \sum_{r=2}^{2^{\ell_0}} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) \qquad (14)
\end{aligned}$$

In (14) above, $\sum_{r=0}^{1} \sum_{h=0}^{2^{\ell_0}-r} x^h y^{2^{\ell_0}-r} \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1) = 0$. The minimum value of r_1 or $r-r_1$ is 1. The maximum value for r_1 or $r-r_1$ such that $x^h y^{2^{\ell_0}-r}$ will have a non-zero coefficient $N_{\ell_0-1}(r_1,h_1) \times N_{\ell_0-1}(r-r_1,h-h_1)$ is 2^{ℓ_0-1}.

Hence, $C_{\ell_0}(x,y) = Y_{\ell_0}(x,y)$.

Let $A_{\ell_0-1}(x,y) = \left(\sum_{r=1}^{2^{\ell_0-1}} \sum_{h=0}^{2^{\ell_0-1}-r} N_{\ell_0-1}(r,h) x^h y^{2^{\ell_0-1}-r} \right)$. It can be easily seen that

$$\begin{aligned}
X_{\ell_0-1}(x,y) &= \sum_{r=0}^{2^{\ell_0-1}} \sum_{h=0}^{2^{\ell_0-1}-r} N_{\ell_0-1}(r,h) x^h y^{2^{\ell_0-1}-r} \\
&= \sum_{h=0}^{2^{\ell_0-1}} N_{\ell_0-1}(0,h) x^h y^{2^{\ell_0-1}} + \sum_{r=1}^{2^{\ell_0-1}} \sum_{h=0}^{2^{\ell_0-1}-r} N_{\ell_0-1}(r,h) x^h y^{2^{\ell_0-1}-r} \\
&= x y^{2^{\ell_0-1}} + A_{\ell_0-1}(x,y) \qquad (15)
\end{aligned}$$

Putting the value of $A_{\ell_0-1}(x,y)$ from (15) and the value of Y_{ℓ_0} from (14) into (12), we get:

$$\begin{aligned}
Y_{\ell_0}(x,y) &= X_{\ell_0-1}^2(x,y) - 2xy^{2^{\ell_0-1}} X_{\ell_0-1}(x,y) - x^2 y^{2^{\ell_0}} \\
&= \left(X_{\ell_0-1}(x,y) - xy^{2^{\ell_0-1}} \right)^2 . \qquad (16)
\end{aligned}$$

Now, to find another relation between the generating functions $X_{\ell_0}(x,y)$ and $Y_{\ell_0}(x,y)$, we multiply both sides of (7) with $x^h y^{2^{\ell_0}-r}$ and sum both sides over $2 \le r \le 2^{\ell_0}$ and $0 \le h \le 2^{\ell_0}$:

$$\sum_{r=2}^{2^{\ell_0}} \sum_{h=1}^{2^{\ell_0}-r} N_{\ell_0}(r,h) x^h y^{2^{\ell_0}-r} = \sum_{r=2}^{2^{\ell_0}} \sum_{h=1}^{2^{\ell_0}-r} T_{\ell_0}(r,h) x^h y^{2^{\ell_0}-r} + \sum_{r=2}^{2^{\ell_0}} \sum_{h=1}^{2^{\ell_0}-r} \sum_{\ell=1}^{\ell_0-1} \left(2^{\ell_0-\ell} x^h y^{2^{\ell_0}-r} \times T_{\ell}(r,h-1) \right). \qquad (17)$$

Adding the values of $N_{\ell_0}(r,h)$ and $T_{\ell_0}(r,h)(=0)$ for $r<2$ and $h\geq 1$ to both sides of the (17) above, we get:

$$\sum_{r=0}^{2^{\ell_0}}\sum_{h=1}^{2^{\ell_0}-r} N_{\ell_0}(r,h)x^h y^{2^{\ell_0}-r} = xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + \sum_{r=0}^{2^{\ell_0}}\sum_{h=1}^{2^{\ell_0}-r} T_{\ell_0}(r,h)x^h y^{2^{\ell_0}-r} + \sum_{r=0}^{2^{\ell_0}}\sum_{h=1}^{2^{\ell_0}-r}\sum_{\ell=1}^{\ell_0-1}\left(2^{\ell_0-\ell}x^h y^{2^{\ell_0}-r}\times T_\ell(r,h-1)\right).$$

Since for $h=0$, $N_{\ell_0}(2^{\ell_0},0)=1$ ($T_{\ell_0}(2^{\ell_0},0)=1$) and for any $r<2^{\ell_0}$, $N_{\ell_0}(r,0)=0$ ($T_{\ell_0}(r,0)=0$),

$$\begin{aligned} X_{\ell_0}(x,y)-1 &= xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + Y_{\ell_0}(x,y) - 1 + \sum_{\ell=1}^{\ell_0-1}\left(2^{\ell_0-\ell}\times\sum_{r=0}^{2^{\ell_0}}\sum_{h=1}^{2^{\ell_0}-r} T_\ell(r,h-1)x^h y^{2^{\ell_0}-r}\right) \\ &= xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + Y_{\ell_0}(x,y) - 1 + \sum_{\ell=1}^{\ell_0-1}\left(2^{\ell_0-\ell}xy^{2^{\ell_0}-2^\ell}\times\sum_{r=0}^{2^\ell}\sum_{h=0}^{2^{\ell_0}-r-1} T_\ell(r,h)x^h y^{2^\ell-r}\right). \quad (18)\end{aligned}$$

Since $2^{\ell_0}-r-1>2^\ell-r$ for $1\leq\ell\leq\ell_0-1$, hence we get:

$$X_{\ell_0}(x,y) = xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + Y_{\ell_0}(x,y) + \sum_{\ell=1}^{\ell_0-1}\left(2^{\ell_0-\ell}xy^{2^{\ell_0}-2^\ell}\times Y_\ell(x,y)\right). \quad (19)$$

From (16) and (19) above, we get:

$$X_{\ell_0}(x,y) = \left(X_{\ell_0-1}(x,y) - xy^{2^{\ell_0-1}}\right)^2 + xy^{2^{\ell_0}} + 2^{\ell_0}x^2y^{2^{\ell_0}-1} + \sum_{\ell=1}^{\ell_0-1}\left(2^{\ell_0-\ell}xy^{2^{\ell_0}-2^\ell}\times\left(X_{\ell-1}(x,y)-xy^{2^{\ell-1}}\right)^2\right). \quad (20)$$

□

A similar generating function was found by Park and Blake in [PB06]. It was derived based on the structural properties of the tree used in creating the subsets of the subset difference method. We have taken a different approach of first finding the recurrence relations for the sequence $N(n,r,h)$ and then deriving the generating function from it. Our generating function is of a slightly different form.

5 Expected Header Length in the CSD method

Given the number of users n ($2^{\ell_1}<n\leq 2^{\ell_1+1}$), and the number of revoked users r, there are $\binom{n}{r}$ possible revocation patterns. Each such revocation pattern gives rise to a subset cover for the privileged users and hence a header in the ciphertext C. We now find the expected value of the header length h for a given n and r.

5.1 Basic Analysis

The Random Experiment: *We consider the random experiment where r out of the n initially un-revoked leaves of the tree $\mathcal{T}^0$ are chosen uniformly at random without replacement and revoked.* This gives rise to an (n,r)-revocation pattern and hence a corresponding subset cover S_c and its header length h. Let $X_{n,r}$ be the random variable taking the value of the header length h due to the (n,r)-revocation pattern of the above experiment. Next, we associate a random variable with each node of the tree $\mathcal{T}^0$. Let $X^i_{n,r}\in\{0,1\}$ be a random variable

associated with node i of $\mathcal{T}^0$. ($X^0_{n,r}$ is associated with the root of $\mathcal{T}^0$.) $X^i_{n,r} = 1$ denotes the event that the cover contains a subset $S_{i,j}$ ($\mathcal{T}^i \setminus \mathcal{T}^j$) where j is some node in the subtree $\mathcal{T}^i$ (j is a descendant of i). In other words, when $X^i_{n,r} = 1$ we say that node i generates a subset for the cover. Similarly, $X^i_{n,r} = 0$ denotes the event that there is no subset $S_{i,j}$ in the cover (i does not generate a subset for the cover). $X^i_{n,r}$ for node i (represented by (ℓ_i, k_i)) will also be written as $X^{\ell_i,k_i}_{n,r}$.

The Expected Header Length: We have $X_{n,r} = X^0_{n,r} + X^1_{n,r} + \ldots + X^{n-2}_{n,r}$. By linearity of expectation:

$$E[X_{n,r}] = E[X^0_{n,r}] + E[X^1_{n,r}] + \ldots + E[X^{n-2}_{n,r}]. \tag{21}$$

Since all the random variables $X^t_{n,r}$ follow *Bernoulli distribution with probability* $\Pr[X^t_{n,r} = 1]$, we get:

$$E[X_{n,r}] = \Pr[X^0_{n,r} = 1] + \Pr[X^1_{n,r} = 1] + \ldots + \Pr[X^{n-2}_{n,r} = 1]. \tag{22}$$

Recall that $\mathcal{P}^0$ is the unique path from the root to a leaf node which contains the nodes at which the non-full subtrees of $\mathcal{T}^0$ are rooted. As we had discussed before, the subtrees $\mathcal{T}^i$ for which i is not on $\mathcal{P}^0$ are full. For a level ℓ of $\mathcal{T}^0$ the subtrees to the left of $\mathcal{P}^0$ are identical in structure (they are all full and have equal number of leaves). Hence, $\Pr[X^i_{n,r} = 1]$ needs to be computed only once for every such node i to the left of $\mathcal{P}^0$ at level ℓ. Similarly for nodes to the right of $\mathcal{P}^0$. Hence, efficient computation of $E[X_{n,r}]$ using (22), boils down to finding $\Pr[X^j_{n,r} = 1]$ level-wise. There are q_ℓ (internal) nodes at all levels $\ell \geq 2$. At level 1, there are $n - 2^{\ell_1}$ ($= q_0/2$) internal nodes. The other $q_1 - (n - 2^{\ell_1})$ nodes at level 1 are leaves. Hence, (22) can also be written as:

$$E[X_{n,r}] = \sum_{\ell=2}^{\ell_0} \sum_{k=1}^{q_\ell} \Pr[X^{\ell,k}_{n,r} = 1] + \sum_{k=1}^{q_0/2} \Pr[X^{1,k}_{n,r} = 1]. \tag{23}$$

When $r = 0$, there is only one set $\mathcal{N}$ in the cover $\mathcal{S}_c$. Hence, $E[X_{n,r}] = 1$. Here on, we will consider $r \geq 1$.

$\Pr[X^0_{n,r} = 1]$ ($= \Pr[X^{\ell_0,1}_{n,r} = 1]$) **for the root node:** We start with finding the probability that the root node generates a subset $S_{0,j}$ for the cover $\mathcal{S}_c$. We define events R^0_{lt} and R^0_{rt} with respect to our random experiment defined above. R^0_{lt} (respectively R^0_{rt}) is the event that there is at least one revoked user in the left (respectively right) subtree of $\mathcal{T}^0$. Hence, $\overline{R^0_{lt}}$ (respectively $\overline{R^0_{rt}}$) is the event that there is no revoked user in the left (respectively right) subtree of $\mathcal{T}^0$. The event $X^0_{n,r} = 1$ occurs when exactly one of the two subtrees of $\mathcal{T}^0$ contains *all* the r revoked users. If all the revoked users are in the left subtree, then the event $\overline{R^0_{rt}}$ occurs and similarly if all the revoked users are in the right subtree the event $\overline{R^0_{lt}}$ occurs. Further, for $r \geq 1$, the events $\overline{R^0_{lt}}$ and $\overline{R^0_{rt}}$ are disjoint. Hence, $\Pr[X^0_{n,r} = 1] = \Pr[\overline{R^0_{lt}}] + \Pr[\overline{R^0_{rt}}]$.

To simplify the computation of these probabilities, we define a new notation $\eta_r(\alpha, \beta)$ to indicate *the probability of choosing r elements from a set of α elements such that β out of these α elements are never chosen.* So, if $\beta \geq \alpha - r + 1$, then $\eta_r(\alpha, \beta) = 0$ by definition. Else, for $0 < \beta < \alpha - r + 1$,

$$\eta_r(\alpha, \beta) = \left(1 - \frac{\beta}{\alpha}\right)\left(1 - \frac{\beta}{\alpha - 1}\right)\left(1 - \frac{\beta}{\alpha - 2}\right) \ldots \left(1 - \frac{\beta}{\alpha - r + 1}\right) = \frac{(\alpha - \beta)_r}{(\beta)_r}. \tag{24}$$

With this definition of $\eta_r(\alpha, \beta)$ in mind, we proceed to compute $\Pr[\overline{R^0_{lt}}]$. In the above random experiment, let Q^0_k be the event that no user from the left subtree of $\mathcal{T}^0$ (having λ_1 leaves) is chosen in the k^{th} trial (out of the r

trials). One can see that $\Pr[R^0_{lt} = 0] = \Pr[Q^0_1 \wedge Q^0_2 \wedge \ldots \wedge Q^0_r]$ and $\Pr[Q^0_k | Q^0_{k-1} \wedge \ldots \wedge Q^0_1] = (1 - \frac{\lambda_1}{n-k+1})$. Hence, we get:

$$\begin{aligned}\Pr[\overline{R^0_{lt}}] &= \Pr[Q^0_1 \wedge Q^0_2 \wedge \ldots \wedge Q^0_r] \\ &= \Pr[Q^0_1] \times \Pr[Q^0_2|Q^0_1] \times \Pr[Q^0_3|Q^0_2 \wedge Q^0_1] \times \Pr[Q^0_r|Q^0_{r-1} \wedge \ldots \wedge Q^0_1] \\ &= \left(1 - \frac{\lambda_1}{n}\right)\left(1 - \frac{\lambda_1}{n-1}\right)\left(1 - \frac{\lambda_1}{n-2}\right)\ldots\left(1 - \frac{\lambda_1}{n-r+1}\right) \\ &= \eta_r(n, \lambda_1) \end{aligned} \tag{25}$$

where λ_1 is the number of nodes in the left subtree of $\mathcal{T}^0$. Similarly, for the right subtree of $\mathcal{T}^0$, we get $\Pr[\overline{R^0_{rt}}] = \eta_r(n, \lambda_2)$ where λ_2 is the number of nodes in the right subtree of $\mathcal{T}^0$. Consequently,

$$\Pr[X^0_{n,r} = 1] = \eta_r(n, \lambda_1) + \eta_r(n, \lambda_2). \tag{26}$$

$\Pr[X^i_{n,r}(= X^{\ell_i,k_i}_{n,r}) = 1]$ for the node i of $\mathcal{T}^i$: Next, we find the probability that node i generates a subset $S_{i,j}$ for the cover. This event $X^i_{n,r} = 1$ occurs when the sibling subtree $\mathcal{T}^s$ of i ($\mathcal{T}^s = \mathcal{T}^{i-1}$ or $\mathcal{T}^{i+1}$) has at least one revoked node and exactly one of the subtrees of i has at least one revoked user. We define the events R^i_{sb}, R^i_{lt} and R^i_{rt} for node i with respect to our random experiment. R^i_{sb} denotes the event that the number of revoked nodes in the sibling subtree of $\mathcal{T}^i$ is non-zero. R^i_{lt} (respectively R^i_{rt}) denotes the event that the number of revoked nodes in the left (respectively right) subtree $\mathcal{T}^{2i+1}$ (respectively $\mathcal{T}^{2i+2}$) is non-zero.

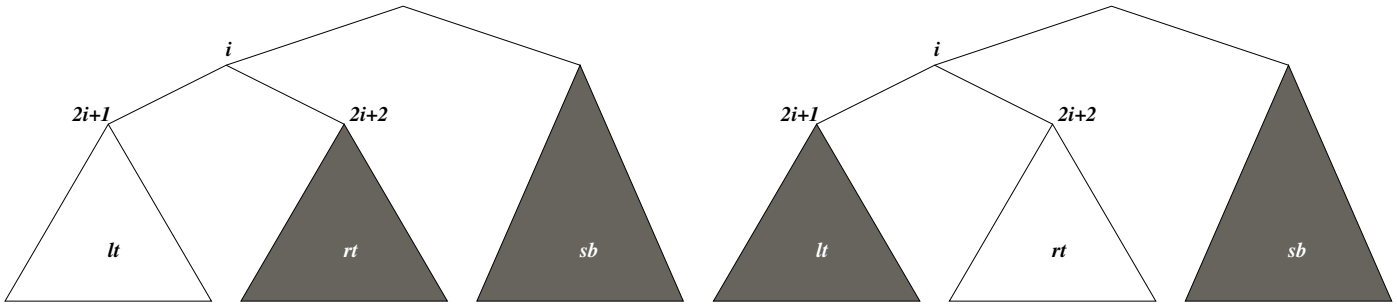

Figure 4: Figures demonstrating the events $R^i_{sb} \wedge R^i_{rt} \wedge \overline{R^i_{lt}}$ and $R^i_{sb} \wedge R^i_{lt} \wedge \overline{R^i_{rt}}$ respectively. The triangles represent subtrees rooted at the respective nodes. White denotes that the subtree has no revoked user in it. Gray denotes that the subtree has at least one revoked user in it. (The sizes of the subtrees are not up to the scale of the number of users in them.)

Lemma 11. *For an internal non-root node i in $\mathcal{T}^0$, the probability that the cover S_c contains a set of the form $\mathcal{T}^i \setminus \mathcal{T}^j$ (where j is some node in the subtree $\mathcal{T}^i$), is given by $\Pr[X^i_{n,r} = 1]$ where*

$$\Pr[X^i_{n,r} = 1] = \Pr[R^i_{sb} \wedge R^i_{rt} \wedge \overline{R^i_{lt}}] + \Pr[R^i_{sb} \wedge R^i_{lt} \wedge \overline{R^i_{rt}}].$$

Proof. For a subset $S_{i,j}$ to occur in the cover (where j is some node in the subtree $\mathcal{T}^i$), there should be at least one revoked user in exactly one of the subtrees ($\mathcal{T}^{2i+1}$ or $\mathcal{T}^{2i+2}$) of the node i. The sibling subtree $\mathcal{T}^s$ should also have at least one revoked user. Hence the event $X^i_{n,r} = 1$ can be divided into two mutually exclusive and exhaustive events. First, when the sibling subtree and the right subtree of $\mathcal{T}^i$ have at least one revoked user in each and the left subtree does not have any: $(R^i_{sb} \wedge R^i_{rt} \wedge \overline{R^i_{lt}})$. Second, when the sibling subtree and the left subtree of $\mathcal{T}^i$ have at least one revoked user in each and the right subtree does not have any: $(R^i_{sb} \wedge R^i_{lt} \wedge \overline{R^i_{rt}})$. Hence the lemma. □

Theorem 12. *For an internal non-root node i of $\mathcal{T}^0$ whose sibling subtree has λ_s leaves,*

$$\begin{aligned}\Pr[X^i_{n,r}=1] &= \eta_r(n,\lambda_{2i+1})+\eta_r(n,\lambda_{2i+2})-\eta_r(n,\lambda_s+\lambda_{2i+1})-\eta_r(n,\lambda_s+\lambda_{2i+2})\\ &\quad -2\eta_r(n,\lambda_{2i+1}+\lambda_{2i+2})+2\eta_r(n,\lambda_s+\lambda_{2i+1}+\lambda_{2i+2}).\end{aligned} \tag{27}$$

For the root node 0 of $\mathcal{T}^0$,

$$\Pr[X^0_{n,r}=1]=\eta_r(n,\lambda_1)+\eta_r(n,\lambda_2). \tag{28}$$

Proof. Let us look at $\Pr[R^i_{sb}\wedge R^i_{rt}\wedge\overline{R^i_{lt}}]$ from Lemma 11.

$$\begin{aligned}\Pr[R^i_{sb}\wedge R^i_{rt}\wedge\overline{R^i_{lt}}] &= \Pr[R^i_{sb}\wedge R^i_{rt}|\overline{R^i_{lt}}]\times\Pr[\overline{R^i_{lt}}]\\ &= \Big(1-\Pr[\overline{R^i_{sb}\wedge R^i_{rt}}|\overline{R^i_{lt}}]\Big)\times\Pr[\overline{R^i_{lt}}]\\ &= \Big(1-\Pr[\overline{R^i_{sb}}|\overline{R^i_{lt}}]-\Pr[\overline{R^i_{rt}}|\overline{R^i_{lt}}]+\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{rt}}|\overline{R^i_{lt}}]\Big)\times\Pr[\overline{R^i_{lt}}]\\ &= \Pr[\overline{R^i_{lt}}]-\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{lt}}]-\Pr[\overline{R^i_{rt}}\wedge\overline{R^i_{lt}}]+\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{rt}}\wedge\overline{R^i_{lt}}].\end{aligned} \tag{29}$$

It can be verified that (29) holds even if $\Pr[\overline{R^i_{lt}}]=0$. Similarly, $\Pr[R^i_{sb}\wedge R^i_{lt}\wedge\overline{R^i_{rt}}]$ of Lemma 11 can be written as:

$$\Pr[R^i_{sb}\wedge R^i_{lt}\wedge\overline{R^i_{rt}}]=\Pr[\overline{R^i_{rt}}]-\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{rt}}]-\Pr[\overline{R^i_{lt}}\wedge\overline{R^i_{rt}}]+\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{lt}}\wedge\overline{R^i_{rt}}]. \tag{30}$$

Next, we deduce the expression for finding $\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{lt}}\wedge\overline{R^i_{rt}}]$ in terms of $\eta_r(\cdot,\cdot)$. Let Q^i_k be the event that no user from the left, right or sibling subtrees of $\mathcal{T}^i$ is chosen in the k^{th} trial (out of the r trials) of the random experiment. One can see that $\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{lt}}\wedge\overline{R^i_{rt}}]=\Pr[Q^i_1\wedge Q^i_2\wedge\ldots\wedge Q^i_r]$ and $\Pr[Q^i_k|Q^i_{k-1}\wedge\ldots\wedge Q^i_1]=\Big(1-\frac{\lambda_s+\lambda_{2i+1}+\lambda_{2i+2}}{n-k+1}\Big)$. Consequently, $\Pr[\overline{R^i_{sb}}\wedge\overline{R^i_{lt}}\wedge\overline{R^i_{rt}}]=\eta_r(n,\lambda_s+\lambda_{2i+1}+\lambda_{2i+2})$. The other probabilities on the right hand sides of (29) and (30) can be found similarly by defining the corresponding event Q^i_k to exclude the users in the respective subtrees from being chosen in the k^{th} trial of the random experiment. From Lemma 11, and substituting the probabilities on the right hand sides of (29) and (30) with their corresponding $\eta_r(\cdot,\cdot)$ equivalents, we get:

$$\begin{aligned}P^i_{n,r}\triangleq\Pr[X^i_{n,r}=1] &= \eta_r(n,\lambda_{2i+1})+\eta_r(n,\lambda_{2i+2})-\eta_r(n,\lambda_s+\lambda_{2i+1})-\eta_r(n,\lambda_s+\lambda_{2i+2})\\ &\quad -2\eta_r(n,\lambda_{2i+1}+\lambda_{2i+2})+2\eta_r(n,\lambda_s+\lambda_{2i+1}+\lambda_{2i+2}).\end{aligned} \tag{31}$$

For the root node, $\Pr[X^0_{n,r}=1]=\Pr[\overline{R^0_{lt}}]+\Pr[\overline{R^0_{rt}}]$ where $\Pr[\overline{R^0_{lt}}]=\eta_r(n,\lambda_1)$ and $\Pr[\overline{R^0_{rt}}]=\eta_r(n,\lambda_2)$. Hence,

$$P^0_{n,r}\triangleq\Pr[X^0_{n,r}=1]=\eta_r(n,\lambda_1)+\eta_r(n,\lambda_2). \tag{32}$$

□

5.2 The Algorithm

Computing $E[X_{n,r}]$: Now that we have the expressions to find $\Pr[X^i_{n,r}=1]$ for all $i\in\{0,\ldots,n-2\}$ in Theorem 12, the values for λ_s, λ_{2i+1} and λ_{2i+2} for node i have to substituted appropriately in (31) and (32). By doing these substitutions for nodes at each level $\ell\in\{1,\ldots,\ell_0\}$ of $\mathcal{T}^0$, we get Algorithm 1. For level $\ell\in\{2,\ldots,\ell_0-1\}$, this computation is done in four steps: (1) for the node $k^{\mathcal{P}}_\ell$ of level ℓ, (2) its sibling subtree, (3) all subtrees (full) to the left of the above two subtrees, and (4) all subtrees (full) to the right of the two subtrees in 1 and 2. The subtree at position $k^{\mathcal{P}}_\ell$ at level ℓ is the only possible non-full subtree for level ℓ and is of height ℓ. If $k^{\mathcal{P}}_\ell$ is odd, its sibling (full) subtree is of height $\ell-1$. If $k^{\mathcal{P}}_\ell$ is even, its sibling (full) subtree is of height ℓ. The subtree at node $k^{\mathcal{P}}_{\ell-1}$ of level $\ell-1$ is always a subtree of the tree rooted at node $k^{\mathcal{P}}_\ell$ of level ℓ. When $k^{\mathcal{P}}_{\ell-1}$ is odd, the right subtree of the tree rooted at node $k^{\mathcal{P}}_\ell$ of level ℓ is full. When $k^{\mathcal{P}}_{\ell-1}$ is even, the left subtree of the tree rooted at node $k^{\mathcal{P}}_\ell$ of level ℓ is full. For the root node 0 and the nodes at level 1, the substitutions are more simple.

n	r	$E[X_{n,r}]$	$\frac{E[X_{n,r}]}{r}$
10^6	2	2.5	1.250
10^6	10	12.48	1.248
10^6	100	124.54	1.248
10^6	1000	1243.77	1.244
10^6	10000	12313.53	1.231

Table 7: $\frac{E[X_{n,r}]}{r}$ for given n and r computed using Algorithm 1 for the CSD scheme.

Time and Space Complexity: To analyze the running time of the algorithm, we observe that each computation of $\eta_r(\alpha, \beta)$ involves $O(r)$ multiplications and there are a constant number of computations of $\eta_r(\alpha, \beta)$ for each level of the tree. Hence, Algorithm 1 requires $O(r \log n)$ multiplications. Algorithm 1 runs using $O(1)$ space. Table 7 lists some values of $E[X_{n,r}]$ for $n = 10^6$ and different values of r.

5.3 Asymptotic Analysis Of $E[X_{n,r}]$ for Full Binary trees

For $n = 2^{\ell_0}$, for any internal node $i \in \{0, \ldots, n-2\}$, $\lambda_{2i+1} = \lambda_{2i+2} = 2^{\ell_i - 1}$. For any node at level $\ell_i > 0$, $\lambda_s = 2^{\ell_i + 1}$. Substituting these values for a node (ℓ, k), (31) becomes:

$$\Pr[X_{n,r}^{\ell,k} = 1] = 2[\eta_r(n, 2^{\ell-1}) - \eta_r(n, 2 \times 2^{\ell-1}) - \eta_r(n, 3 \times 2^{\ell-1}) + \eta_r(n, 4 \times 2^{\ell-1})]. \tag{33}$$

This probability is independent of k. In other words, the probability of generating a subset for the cover is equal for all nodes at level ℓ. Hence, we define the following:

Definition 2. *$B_{n,r}^{(\ell)}$: Let $\ell (1 \leq \ell \leq \ell_0)$ be a level number of the tree $\mathcal{T}^0$ and $n = 2^{\ell_0}$. $B_{n,r}^{(\ell)}$ is defined as $\Pr[X_{n,r}^{\ell,k} = 1]$ of (33) for the node (ℓ, k) of $\mathcal{T}^0$. Hence,*

$$B_{n,r}^{(\ell)} = 2[\eta_r(n, 2^{\ell-1}) - \eta_r(n, 2 \times 2^{\ell-1}) - \eta_r(n, 3 \times 2^{\ell-1}) + \eta_r(n, 4 \times 2^{\ell-1})].$$

Note that by this definition, for the only node (the root node) at level ℓ_0, $B_{n,r}^{(\ell_0)} = 2\eta_r(n, 2^{\ell_0 - 1})$ which is consistent with (32) for $n = 2^{\ell_0}$. Hence, we define the following:

Definition 3. *$H_{n,r}$: For a given $n = 2^{\ell_0}$ and r, the expected header length $H_{n,r}$ due to the subset cover algorithm of the CSD scheme is defined as:*

$$H_{n,r} = E[X_{n,r}] = \sum_{\ell=1}^{\ell_0} 2^{\ell_0 - \ell} B_{n,r}^{(\ell)}.$$

Definition 4. *$D_{n,r}$: For a given $n = 2^{\ell_0}$, the difference between the expected header lengths for the number of revoked users being r and $r-1$ is defined as $D_{n,r}$. Hence,*

$$D_{n,r} = H_{n,r} - H_{n,r-1}.$$

Algorithm 1 Find $E[X_{n,r}]$ for the CSD scheme.

Require: $n \geq 1$; $0 \leq r \leq n$;
Function $\eta_r(\cdot,\cdot)$ is defined in (24);
Function $P^i_{n,r}(\lambda_{2i+1}, \lambda_{2i+2}, \lambda_s)$ is defined in (31).
Ensure: Output $E[X_{n,r}]$ computed in *result*.
Compute ℓ_0 such that $2^{\ell_0-1} < n \leq 2^{\ell_0}$
$q_0 \leftarrow 2 \times (n - 2^{\ell_0-1})$ {The number of leaves at level 0}
{For the root node 0:}
if $q_0 \leq 2^{\ell_0-1}$ **then**
$result \leftarrow \eta_r(n, n - 2^{\ell_0-2}) + \eta_r(n, 2^{\ell_0-2})$
else
$result \leftarrow \eta_r(n, 2^{\ell_0-1}) + \eta_r(n, n - 2^{\ell_0-1})$
end if
{For the nodes at levels $\ell \in \{1, \ldots, \ell_1\}$:}
for $\ell = (\ell_0 - 1)$ down to 2 **do**
$q_\ell \leftarrow 2^{\ell_0-\ell}$ {The number of nodes at level ℓ}
$k^{\mathcal{P}}_\ell \leftarrow \lceil \frac{q_0}{2^\ell} \rceil$ {The position of the unique node at level ℓ on path $\mathcal{P}^0$}
$\lambda^{\ell,\mathcal{P}} \leftarrow n - (k^{\mathcal{P}}_\ell - 1) \times 2^\ell - (q_\ell - k^{\mathcal{P}}_\ell) \times 2^{\ell-1}$ {# leaves in the only possibly non-full subtree of level ℓ}
if $k^{\mathcal{P}}_\ell$ is odd **then**
$result \leftarrow result + (k^{\mathcal{P}}_\ell - 1) \times P^i_{n,r}(2^{\ell-1}, 2^{\ell-1}, 2^\ell)$
if $k^{\mathcal{P}}_{\ell-1}$ is odd **then**
$result \leftarrow result + P^i_{n,r}(\lambda^{\ell,\mathcal{P}} - 2^{\ell-2}, 2^{\ell-2}, 2^{\ell-1})$
else
$result \leftarrow result + P^i_{n,r}(2^{\ell-1}, \lambda^{\ell,\mathcal{P}} - 2^{\ell-1}, 2^{\ell-1})$
end if
$result \leftarrow result + P^i_{n,r}(2^{\ell-2}, 2^{\ell-2}, \lambda^{\ell,\mathcal{P}})$
$result \leftarrow result + (q_\ell - (k^{\mathcal{P}}_\ell + 1)) \times P^i_{n,r}(2^{\ell-2}, 2^{\ell-2}, 2^{\ell-1})$
else
$result \leftarrow result + (k^{\mathcal{P}}_\ell - 2) \times P^i_{n,r}(2^{\ell-1}, 2^{\ell-1}, 2^\ell)$
$result \leftarrow result + P^i_{n,r}(2^{\ell-1}, 2^{\ell-1}, \lambda^{\ell,\mathcal{P}})$
if $k^{\mathcal{P}}_{\ell-1}$ is odd **then**
$result \leftarrow result + P^i_{n,r}(\lambda^{\ell,\mathcal{P}} - 2^{\ell-2}, 2^{\ell-2}, 2^\ell)$
else
$result \leftarrow result + P^i_{n,r}(2^{\ell-1}, \lambda^{\ell,\mathcal{P}} - 2^{\ell-1}, 2^\ell)$
end if
$result \leftarrow result + (q_\ell - k^{\mathcal{P}}_\ell) \times P^i_{n,r}(2^{\ell-2}, 2^{\ell-2}, 2^{\ell-1})$
end if
end for
{For the nodes at level $\ell = 1$:}
if $\frac{q_0}{2}$ is odd **then**
$result \leftarrow result + (\frac{q_0}{2} - 1) \times P^i_{n,r}(1,1,2) + P^i_{n,r}(1,1,1)$
else
$result \leftarrow result + (\frac{q_0}{2}) \times P^i_{n,r}(1,1,2)$
end if
return *result*.

We further observe that:

$$\begin{aligned} H_{n,r} &= H_{n,r-1} + D_{n,r} \\ &= H_{n,r-2} + D_{n,r} + D_{n,r-1} \\ &= H_{n,r-3} + D_{n,r} + D_{n,r-1} + D_{n,r-2} \\ &= \dots \\ &= H_1 + \sum_{i=2}^{r} D_{n,i} \\ &= 1 + \sum_{i=2}^{r} D_{n,i}. \end{aligned} \tag{34}$$

Using the definition of $B_{n,r}^{(\ell)}$ we also get:

$$\begin{aligned} D_{n,r} &= H_{n,r} - H_{n,r-1} \\ &= \sum_{\ell=1}^{\ell_0} 2^{\ell_0-\ell} \left(B_{n,r}^{(\ell)} - B_{n,r-1}^{(\ell)} \right). \end{aligned} \tag{35}$$

In (35), $\eta_r(n,m) - \eta_{r-1}(n,m)$ can be rewritten as follows:

$$\begin{aligned} \eta_r(n,m) - \eta_{r-1}(n,m) &= \frac{(n-m)_r}{(n)_r} - \frac{(n-m)_{r-1}}{(n)_{r-1}} \\ &= \frac{(n-m)(n-m-1)\dots(n-m-r+2)}{n(n-1)\dots(n-r+2)} \times \left(\frac{n-m-r+1}{n-r+1} - 1 \right) \\ &= \frac{(n-m)(n-m-1)\dots(n-m-r+2)}{n(n-1)\dots(n-r+2)} \times \frac{-m}{n-r+1} \\ &= \frac{(n-m)_{r-1}}{(n)_{r-1}} \times \frac{-m}{n-r+1} \\ &= -\eta_{r-1}(n,m) \times \frac{m}{n-r+1}. \end{aligned} \tag{36}$$

Hence from (35) and (36) we get:

$$\begin{aligned} D_{n,r+1} &= \sum_{\ell=1}^{\ell_0} 2^{\ell_0-\ell} \left(B_{n,r+1}^{(\ell)} - B_{n,r}^{(\ell)} \right) \\ &= \frac{2n}{n-r} \sum_{\ell=1}^{\ell_0} \frac{1}{2^\ell} \Big(-2^{\ell-1}\eta_r(n, 2^{\ell-1}) + 2 \times 2^{\ell-1}\eta_r(n, 2 \times 2^{\ell-1}) \\ &\qquad + 3 \times 2^{\ell-1}\eta_r(n, 3 \times 2^{\ell-1}) - 4 \times 2^{\ell-1}\eta_r(n, 4 \times 2^{\ell-1}) \Big) \\ &= \frac{n}{n-r} \left[-\eta_r(n,1) + \eta_r(n,2) + 3\eta_r(n,3) - 3 \sum_{\ell=1}^{\ell_0-1} \left(\eta_r(n, 2 \times 2^\ell) - \eta_r(n, 3 \times 2^\ell) \right) \right]. \end{aligned} \tag{37}$$

Here, we have made use of the fact that $\eta_r(\alpha, \beta) = 0$ when $\beta \geq \alpha - r + 1$. From (37), we calculate the value of

$H_{n,2}$ as follows:

$$\begin{aligned}
H_{n,2} &= H_{n,1} + \frac{n}{n-1}\left[-\eta_1(n,1) + \eta_1(n,2) + 3\eta_1(n,3) - 3\sum_{\ell=1}^{\ell_0-1}\Big(\eta_1(n,2\times 2^\ell) - \eta_1(n,3\times 2^\ell)\Big)\right] \\
&= 1 + \frac{n}{n-1}\left[\frac{3(n-3)}{n} + \frac{n-2}{n} - \frac{n-1}{n} - 3\sum_{\ell=1}^{\ell_0-2}\left(\frac{n-2\times 2^\ell}{n} - \frac{n-3\times 2^\ell}{n}\right)\right] \\
&= 1 + \frac{n}{n-1}\left[\frac{3(n-3)}{n} + \frac{n-2}{n} - \frac{n-1}{n} - \frac{3(n-2)}{2n}\right] \\
&= 1 + \frac{n}{n-1}\left[\frac{3n-14}{2n}\right] \\
&= 1 + \frac{3-\frac{14}{n}}{2(1-\frac{1}{n})}. \qquad (38)
\end{aligned}$$

Note that $\lim_{n\to\infty} H_{n,2} = \frac{5}{2} = 1.25 \times 2$.

Now we analyze $D_{n,r+1}$ in (37) for $r > 2$. We use the notation $x \uparrow a$ to indicate that x increases to a and $x \downarrow a$ to indicate that x decreases to a.

Lemma 13. *$\eta_r(n,3) = \frac{(n-3)_r}{(n)_r} \uparrow 1$ as $n \uparrow \infty$.*

Proof. For any given n, $\frac{(n-3)_r}{(n)_r} < 1$.

$$\begin{aligned}
\lim_{n\to\infty} \eta_r(n,3) &= \lim_{n\to\infty} \frac{(n-3)_r}{(n)_r} \\
&= \lim_{n\to\infty} \frac{(n-3)(n-2)\dots(n-3-r+1)}{n(n-1)\dots(n-r+1)} \\
&= \lim_{n\to\infty} \frac{(1-\frac{3}{n})(1-\frac{2}{n})\dots(1-\frac{r+2}{n})}{(1)(1-\frac{1}{n})\dots(1-\frac{r-1}{n})} \\
&= 1. \qquad (39)
\end{aligned}$$

□

Hence, $3\eta_r(n,3) \uparrow 3$ as $n \uparrow \infty$.

Lemma 14. *$\eta_r(n,2) - \eta_r(n,1) \uparrow 0$ as $n \uparrow \infty$.*

Proof. For any given n, $\eta_r(n,2) - \eta_r(n,1) < 0$.

$$\begin{aligned}
\lim_{n\to\infty} \eta_r(n,2) - \eta_r(n,1) &= \lim_{n\to\infty} \left[\left(1-\frac{2}{n}\right)\dots\left(1-\frac{2}{n-r+1}\right) - \left(1-\frac{1}{n}\right)\dots\left(1-\frac{1}{n-r+1}\right)\right] \\
&= 0. \qquad (40)
\end{aligned}$$

□

Hence, we claim that $(-\eta_r(n,1)+\eta_r(n,2)+3\eta_r(n,3)) \uparrow 3$ as $n \uparrow \infty$. $\frac{n}{n-r} \uparrow 1$ as $n \uparrow \infty$. Finally, we look at $\sum_{\ell=1}^{\ell_0-2}\big(\eta_r(n,2\times 2^\ell - \eta_r(n,3\times 2^\ell)\big)$ to complete the analysis.

$$\begin{aligned}
\sum_{\ell=1}^{\ell_0-2}\Big(\eta_r(n,2\times 2^\ell)-\eta_r(n,3\times 2^\ell)\Big) &= \sum_{\ell=1}^{\ell_0-2}\left(\frac{(n-2\times 2^\ell)_r}{(n)_r}-\frac{(n-3\times 2^\ell)_r}{(n)_r}\right)\\
&\geq \frac{1}{(n)_r}\sum_{\ell=1}^{\ell_0-2}\Big((n-r+1-2\times 2^\ell)^r-(n-3\times 2^\ell)^r\Big)\\
&= \frac{1}{(n)_r}\sum_{\ell=1}^{\ell_0-2}\Big((2^{\ell_0}-2^{\ell+1}-r+1)^r-(2^{\ell_0}-3\times 2^\ell)^r\Big)\\
&= \frac{1}{(n)_r}\sum_{\ell=2}^{\ell_0-1}\Big((2^{\ell_0}-2^{\ell_0-\ell+1}-r+1)^r-(2^{\ell_0}-3\times 2^\ell)^r\Big)\\
&= \frac{1}{(n)_r}\sum_{\ell=2}^{\ell_0-1}\left(\Big(\Big(\frac{2^\ell-2}{2^\ell}\Big)n-r+1\Big)^r-\Big(\Big(\frac{2^\ell-3}{2^\ell}\Big)n\Big)^r\right)\\
&\geq \frac{1}{n^r}\sum_{\ell=2}^{\ell_0-1}\left(\Big(\Big(\frac{2^\ell-2}{2^\ell}\Big)n-r+1\Big)^r-\Big(\Big(\frac{2^\ell-3}{2^\ell}\Big)n\Big)^r\right)\\
&= \sum_{\ell=2}^{\ell_0-1}\left(\Big(\frac{2^\ell-2}{2^\ell}-\frac{r-1}{n}\Big)^r-\Big(\frac{2^\ell-3}{2^\ell}\Big)^r\right). \qquad (41)
\end{aligned}$$

We define K_r as follows:

Definition 5. *K_r:*

$$K_r = \lim_{n\to\infty}\sum_{\ell\geq 2}\left(\Big(\frac{2^\ell-2}{2^\ell}-\frac{r-1}{n}\Big)^r-\Big(\frac{2^\ell-3}{2^\ell}\Big)^r\right)$$

Hence,

$$\begin{aligned}
K_r &= \lim_{n\to\infty}\sum_{\ell\geq 2}\left(\Big(\frac{2^\ell-2}{2^\ell}-\frac{r-1}{n}\Big)^r-\Big(\frac{2^\ell-3}{2^\ell}\Big)^r\right) &&= \sum_{\ell\geq 2}\left(\Big(\frac{2^\ell-2}{2^\ell}\Big)^r-\Big(\frac{2^\ell-3}{2^\ell}\Big)^r\right)\\
&= \sum_{\ell\geq 2}\frac{1}{2^{r\ell}}\left(\Big(2^\ell-2\Big)^r-\Big(2^\ell-3\Big)^r\right) &&= \sum_{k=1}^{r}(-1)^k\binom{r}{k}(2^k-3^k)\sum_{\ell\geq 2}\frac{2^{\ell(r-k)}}{2^{r\ell}}\\
&= \sum_{k=1}^{r}(-1)^k\binom{r}{k}(2^k-3^k)\sum_{\ell\geq 2}\frac{1}{2^{\ell k}}. \qquad (42)
\end{aligned}$$

Since $\sum_{\ell\geq 2}\frac{1}{2^{\ell k}} = \frac{1}{2^k(2^k-1)}$, we get

$$\begin{aligned}
K_r &= \sum_{k=1}^{r}(-1)^k\binom{r}{k}\frac{(2^k-3^k)}{2^k(2^k-1)}\\
&= \sum_{k=1}^{r}(-1)^k\binom{r}{k}\frac{(2^k-3^k)}{(2^k-1)}-\sum_{k=1}^{r}(-1)^k\binom{r}{k}\frac{(2^k-3^k)}{(2^k)}\\
&= \Big(-\frac{1}{2}\Big)^r+\sum_{k=1}^{r}(-1)^k\binom{r}{k}\frac{(2^k-3^k)}{(2^k-1)}. \qquad (43)
\end{aligned}$$

r	D_r	$\frac{H_r}{r}$
2	$\frac{3}{2}$	1.25
3	$\frac{5}{4}$	1.25
4	$\frac{69}{56}$	1.24553571
5	$\frac{417}{336}$	1.24464285
6	$\frac{25953}{20832}$	1.24483967

Table 8: Ratio $\frac{H_r}{r}$ for different values of r.

r	$n = 200$	$n = 256$	Extra Bytes
10	12	12	0
20	23	23	0
30	32	33	16
40	40	42	32
50	46	50	64

Table 9: The expected header lengths for $n = 200$ and $n = 256$ for different r and the number of extra bytes needed per message of broadcast (assuming each session key is 128-bit long).

We also define:

Definition 6. *H_r and D_r:*

$$H_r = \lim_{n\to\infty} H_{n,r}$$
$$D_r = \lim_{n\to\infty} D_{n,r}$$

The next result summarizes the above analysis.

Theorem 15. *For all $n \geq 1$, $r \geq 1$, the expected header length $H_{n,r} \uparrow H_r$, as n increases through powers of two, where*

$$H_r = 3r - 2 - 3 \times \sum_{i=1}^{r-1} \left(\Big(-\frac{1}{2} \Big)^i + \sum_{k=1}^{i} (-1)^k \binom{i}{k} \frac{(2^k - 3^k)}{(2^k - 1)} \right).$$

Proof. From (34), we get $H_r = 1 + \sum_{i=2}^{r} D_i$. Further, from (37), (41) and (43), we get $D_{r+1} = 3 - 3K_r$ where K_r is given by (43). □

Table 8 lists the values of D_r and $\frac{H_r}{r}$ for small values of r. This table shows the ratio $\frac{H_r}{r}$ is always less than $1.25r$.

In [NNL01], it was proved that for the SD scheme, the expected header length was bounded above by $1.38r$. They also mentioned that simulation results have shown a tighter upper bound of $1.25r$. Our analysis gives theoretical support for this observation.

5.4 Other Experimental Results

We return to the issue of comparing the CSD method to that of the SD method with dummy users. The situation where the dummy users form a block has been discussed in details in Section 3.2. Let us consider the situation

r	$n = 1500$	$n = 2048$	Extra Bytes
50	61	61	0
100	116	118	32
150	167	172	80
200	213	223	160
250	255	270	240
300	293	314	336

Table 10: The expected header lengths for $n = 1500$ and $n = 2048$ for different r and the number of extra bytes needed per message of broadcast (assuming each session key is 128-bit long).

r	$n = 10000$	$n = 16384$	Extra Bytes
500	589	602	208
1000	1109	1162	848
1500	1561	1680	1904
2000	1947	2157	3360
2500	2267	2593	5216
3000	2521	2988	7472

Table 11: The expected header lengths for $n = 10000$ and $n = 16384$ for different r and the number of extra bytes needed per message of broadcast (assuming each session key is 128-bit long).

n	16	17	18	19	20	21	22	23
$\frac{E[X_{n,r}]}{2}$	1.167	1.169	1.180	1.184	1.195	1.200	1.210	1.215
n	24	25	26	27	28	29	30	31
$\frac{E[X_{n,r}]}{2}$	1.225	1.217	1.214	1.209	1.208	1.207	1.208	1.207

Table 12: $\frac{E[X_{n,r}]}{r}$ for $r = 2$, $16 \leq n < 32$ from Algorithm 1 for the CSD scheme.

where the dummy users are randomly distributed. If these are all considered to be revoked, then there is a large penalty on the transmission overhead. This is because the expected header length is linear in the number of revoked users. So, suppose that the randomly distributed dummy users are viewed as being privileged by the cover generation algorithm.

Running Algorithm 1 for different values of n and r we compare the transmission efficiency of the CSD method with the SD method with dummy users. Additionally, we report other observations on the expected header length of the CSD method.

1. For a fixed $n < 2^{\ell_0}$, as r goes above a certain minimum, the expected header length of the CSD method is significantly better than the corresponding instantiation of the SD method. As an example, for $n = 10000$, the expected header length is 1561 for $r = 1500$ while for the corresponding $n = 16384$ of the SD method, the expected header length is 1680 for the same r. Assuming the function F_K used for encrypting each block of digital data is AES-128, this difference of 119 in the expected header length causes an extra bandwidth consumption of 1904 ($= 119 \times 16$) bytes per message on an average. Tables 9, 10 and 11 list the expected header lengths for $n = 200$, 1500 and 10000 and the corresponding next powers of two for different values of r.

2. For $n = 200$, by running the algorithm for computing the expected header length, we observe that the expected header lengths are better compared to $n = 256$ for all $r > 5$. Thus, CSD is more efficient in terms of the transmission overhead efficiency for all $r > 5$ for $n = 200$. Similarly, CSD gains over SD when $n = 1500$ for all $r > 7$ and when $n = 10000$, it gains for all $r > 28$. For real-time scenarios like Pay-TV, $n = 10000$ and $r > 28$ are practical numbers. Thus, the CSD method will provide better transmission efficiency than SD for many practical purposes.

3. For full binary trees, we know from (38) that for $r = 2$, the limiting value of $\frac{E[X_{n,r}]}{2}$ is 1.25. By running our algorithm, we also observe that for n a power of two, the expected header length increases with increasing n for all $r \geq 2$.

4. For $r = 2$, as we keep increasing n from 2^ℓ to $2^{\ell+1} - 1$, the ratio $\frac{E[X_{n,r}]}{r}$ increases almost uniformly to reach a local maximum at $n = 2^\ell + 2^{\ell-1}$ and then decreases. The data in Table 12 demonstrates this behaviour for $16 \leq n < 32$. For $32 \leq n < 64$, the maximum value of $\frac{E[X_{n,r}]}{r}$ is 1.225 observed at $n = 24$ and for $128 \leq n < 256$, the maximum value is 1.271 and is observed at $n = 192$. However, as r increases, the behaviour of the above ratio changes, with local glitches disrupting the uniformity at most places.

6 Conclusion

We have proposed a new BE scheme which extends the tree-based SD scheme of [NNL01]. The new Complete Tree Subset Difference method is capable of accommodating any arbitrary number of users (may not be a power of two) and hence subsumes the SD scheme of [NNL01]. All results of the CSD scheme that we subsequently prove also hold for the SD scheme.

Detailed combinatorial analysis of the CSD scheme is done by finding two recurrences to count the number of ways r out of n users can be revoked to result in a subset cover size of h in the CSD method. Using these recurrences, it is proved that the maximum possible header length for a given r is $2r - 1$ (no worse than the SD scheme even though arbitrary number of users are accommodated) and the maximum header length for all r is $\left\lfloor \frac{n}{2} \right\rfloor$. The recurrences are the most efficient tool as per our knowledge to generate exhaustive data for the above count and also the only tool to find and prove the expression for the minimum number of users required to be in a system so that for a given r, the maximum cover size would reach $2r - 1$. For n a power of two, a generating function is found for generating the same sequence as the recurrences.

Probabilistic analysis of the revocation patterns in the CSD scheme gives the most important result of this work: an efficient algorithm to compute the expected header length for a given n and r. Using this algorithm, it is shown that for practical values of n and r, the CSD scheme provides better transmission efficiency as compared to the SD scheme.

References

[AAC] AACS. Advanced access content system, `http://www.aacsla.com`. aacsla.

[AK08] Per Austrin and Gunnar Kreitz. Lower bounds for subset cover based broadcast encryption. In *Proceedings of the Cryptology in Africa 1st international conference on Progress in cryptology*, AFRICACRYPT'08, pages 343–356, Berlin, Heidelberg, 2008. Springer-Verlag.

[AKI03] Nuttapong Attrapadung, Kazukuni Kobara, and Hideki Imai. Sequential key derivation patterns for broadcast encryption and key predistribution schemes. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 374–391. Springer, 2003.

[Ber91] Shimshon Berkovits. How to broadcast a secret. In *EUROCRYPT*, pages 535–541, 1991.

[BS11] Sanjay Bhattacherjee and Palash Sarkar. An analysis of the Naor-Naor-Lotspeich Subset Difference algorithm (for possibly incomplete binary trees). In Daniel Augot and Anne Canteaut, editors, *Workshop on Coding and Cryptography*, pages 483–492, April 11-15, 2011.

[DRM] DRM. Digital rights management, `http://en.wikipedia.org/wiki/DRM\_(computing)`. wikipedia.

[EOPR] Christopher Eagle, Mohamed Omar, Daniel Panario, and Bruce Richmond. Average-case analysis of two revocation schemes for stateless receivers.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002.

[JHC+05] 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 Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 559–574. Springer, 2005.

[LS98] Michael Luby and Jessica Staddon. Combinatorial bounds for broadcast encryption. In *EUROCRYPT*, pages 512–526, 1998.

[MMW09] Thomas Martin, Keith M. Martin, and Peter R. Wild. Establishing the broadcast efficiency of the subset difference revocation scheme. *Des. Codes Cryptography*, 51(3):315–334, 2009.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[PB06] E. C. Park and Ian F. Blake. On the mean number of encryptions for tree-based broadcast encryption schemes. *J. Discrete Algorithms*, 4(2):215–238, 2006.

[PGM04] Carles Padró, Ignacio Gracia, and Sebastià Martín Molleví. Improving the trade-off between storage and communication in broadcast encryption schemes. *Discrete Applied Mathematics*, 143(1-3):213–220, 2004.

[PGMM02] Carles Padró, Ignacio Gracia, Sebastià Martín Molleví, and Paz Morillo. Linear key predistribution schemes. *Designs, Codes and Cryptography*, 25:281–298, 2002. 10.1023/A:1014939630572.

[PGMM03] Carles Padró, Ignacio Gracia, Sebastià Martín Molleví, and Paz Morillo. Linear broadcast encryption schemes. *Discrete Applied Mathematics*, 128(1):223–238, 2003.

[Sti97] Douglas R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Des. Codes Cryptography*, 12(3):215–243, 1997.

[SW98] D. R. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discret. Math.*, 11:41–53, February 1998.

Appendices

A The Subset Difference Method (For Possibly Incomplete Binary Trees)

This paper is a modified and an extended version of our work on the SD Method for possibly incomplete binary trees [BS11]. We proposed a modification to the SD method in order to accommodate arbitrary number of users n (not a power of two). These n users were accommodated in the leaves of an *incomplete* tree. The structure of the tree was such that when $2^t < n \leq 2^{t+1}$, the left subtree of the root node had 2^t leaves and the right subtree had $n - 2^t$ leaves. Hence, the tree was unbalanced.

In [BS11], we had formulated the corresponding recurrences $N(n, r, h)$ and $T(n, r, h)$ for counting the number of (n, r)-revocation patterns that resulted in a header length of h. For an arbitrary n ($2^{t_0} < n \leq 2^{t_0+1}$ for some t_0), $T(n, r, h)$ and $N(n, r, h)$ were given by the following recurrences:

$$T(n, r, h) = \sum_{r_1=1}^{r-1} \sum_{h_1=0}^{h} N(2^{t_0}, r_1, h_1) \times N(n - 2^{t_0}, r - r_1, h - h_1) \tag{44}$$

$$\begin{aligned} N(n, r, h) = T(n, r, h) &+ \sum_{\ell=1}^{t_0} \Big(q_\ell \times T(2^\ell, r, h-1) \Big) \\ &+ \sum_{\ell=1}^{t_0} \left(\left\lfloor \frac{\rho_\ell - 1}{2^{\ell-1}} \right\rfloor \times T(\rho_\ell, r, h-1) \right) \end{aligned} \tag{45}$$

where $q_\ell = \left\lfloor \frac{n}{2^\ell} \right\rfloor$ and $\rho_\ell = n - (q_\ell \times 2^\ell)$. Boundary values remain unaffected by the change in the tree structure.

We had considered the same random experiment as we did in this paper and had computed the probability that a node (ℓ, k) of the tree gives rise to a set for the cover.

$$\begin{aligned} \Pr[X_{n,r}^{\ell,k} = 1] = \eta_r(n, n_\ell) + \eta_r(n, n_r) - \eta_r(n, n_s + n_\ell) - \eta_r(n, n_s + n_r) \\ - 2\eta_r(n, n_\ell + n_r) + 2\eta_r(n, n_s + n_\ell + n_r) \end{aligned} \tag{46}$$

Using this expression to compute the probability for each node and then computing the sum of the probabilities of each internal node in the tree from the following equation, we had obtained the expected header length for the tree structure considered in [BS11].

$$E[X_{n,r}] = \sum_{\ell=1}^{t_0+1} \sum_{k=1}^{q_\ell} \Pr[X_{n,r}^{\ell,k} = 1] + \sum_{\ell=1}^{t_0} \left\lfloor \frac{\rho_\ell - 1}{2^{\ell-1}} \right\rfloor \times \Pr[X_{n,r}^{\ell,q_\ell+1} = 1]. \tag{47}$$

Using this algorithm, we observed that the expected header length took values around $1.5r$ for large n (compared to r) of the form $2^{t_0} + 1$. This was a sharp rise from the observed value of $1.25r$ for the SD scheme. We analyzed and understood that this deviation was because of the unbalanced nature of the tree. This motivated us to work on a balanced tree scheme that we have presented in this paper.